ISSN 2220-3397

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

# Программная инженерия

ПрИн
9
2022
Том 13

Рисунки к статье **С. М. Зуева, Д. О. Варламова, У. И. Акрамова, В. В. Куксы**

# «ПРОГРАММНО РЕАЛИЗОВАННАЯ МОДЕЛЬ УСТРОЙСТВА КОНТРОЛЯ ПАРАМЕТРОВ АВТОМАТИЧЕСКОЙ КОРОБКИ ПЕРЕКЛЮЧЕНИЯ ПЕРЕДАЧ АВТОМОБИЛЯ»

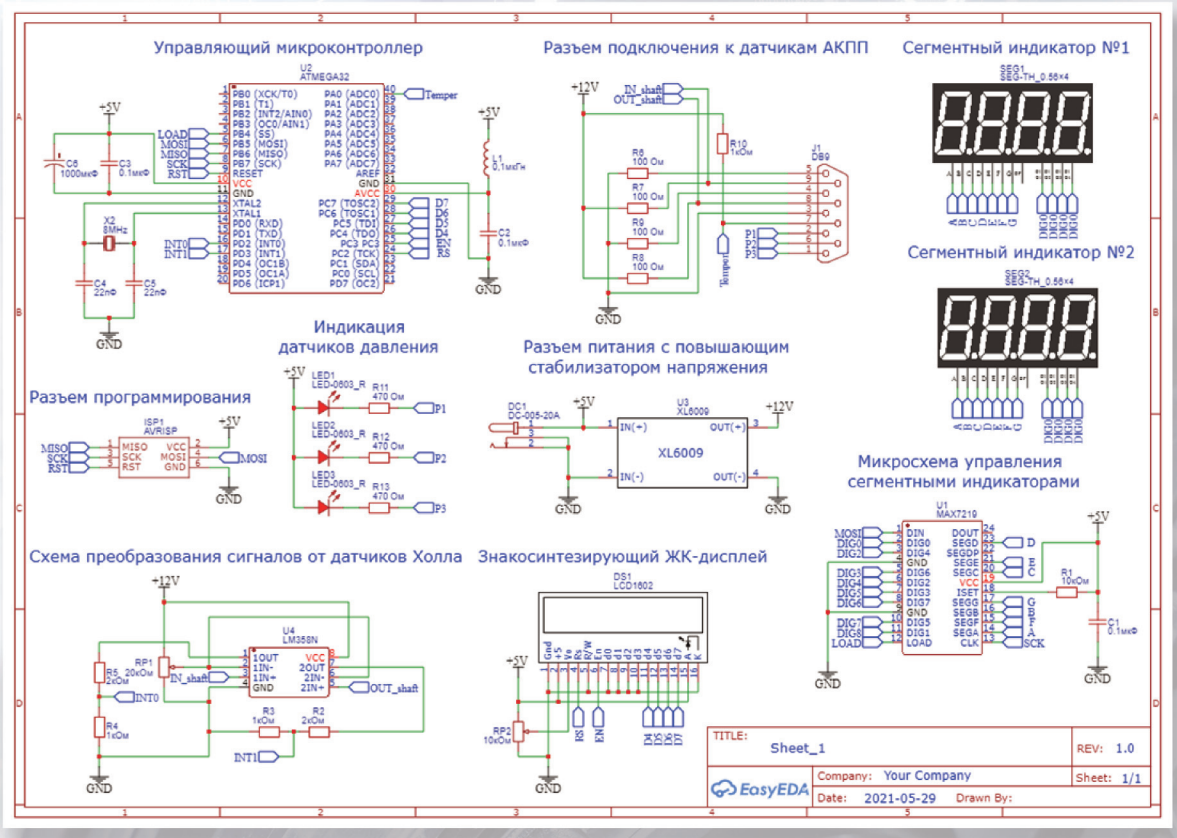


**Рис. 1. Электрическая схема устройства**

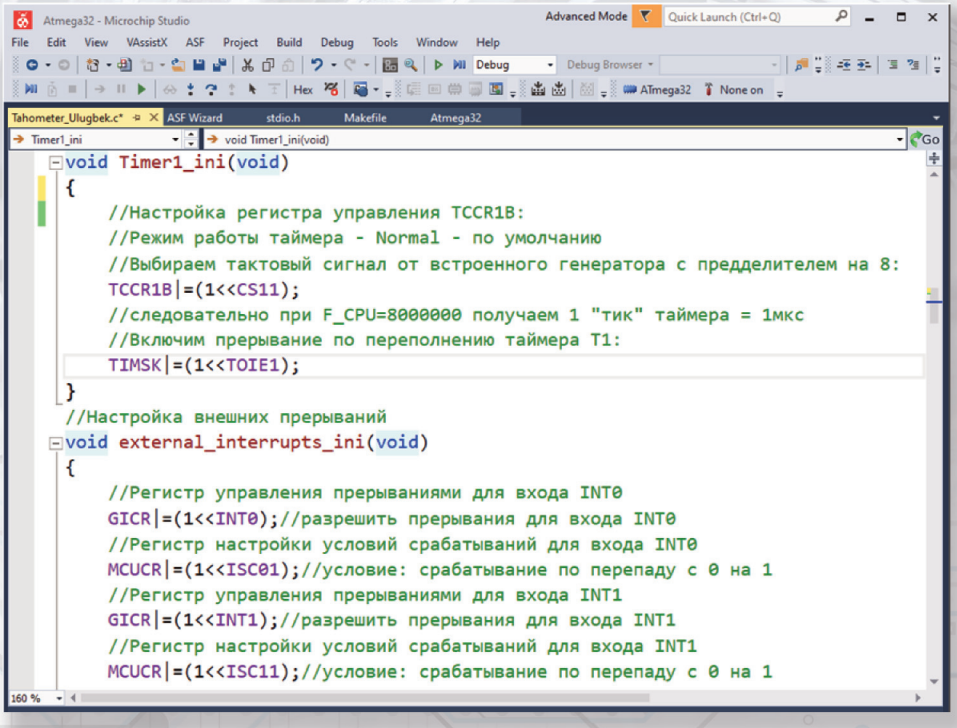


**Рис. 2. Окно программы Microchip Studio**

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ


Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397



Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

# СОДЕРЖАНИЕ



**Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.**

Журнал распространяется по подписке, которую можно оформить в подписных агентствах (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции (для юридических лиц).
Тел.: (499) 270-16-52.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в Российский индекс научного цитирования (РИНЦ) и Russian Science Citation Index (RSCI).
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

# SOFTWARE ENGINEERING

## PROGRAMMNAYA INGENERIA

Vol. 13
N 9
2022

**Published since September 2010** **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**



# CONTENTS



**Information about the jornal is available online at:**
**http://novtex.ru/prin/eng e-mail:prin@novtex.ru**

УДК 519.237.5
DOI: 10.17587/prin.13.415-424

**В. А. Галатенко,** д-р физ.-мат. наук, зав. сектором автоматизации программирования, galat@niisi.ras.ru, **К. А. Костюхин,** канд. физ.-мат. наук, ст. науч. сотр., kost@niisi.ras.ru, Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук, Москва

# Аппаратная отладка: обзор современных подходов*

*Повсеместное распространение сложных устройств, построенных на так называемых системах на кристалле (System on Chip, SoC), с несколькими процессорными ядрами ставит новые задачи перед разработчиками встраиваемых систем. Новые средства разработки, специально предназначенные для сложных систем на кристалле, могут помочь их решить. Однако эти средства, как правило, ограничены функциональностью инструментов поддержки отладки. Высококачественная поддержка отладки с расширенными функциями необходима для использования всех преимуществ сложных SoC-устройств при одновременном сокращении времени разработки. В статье рассмотрены разные механизмы и способы реализации поддержки отладки систем на кристалле, предназначенных для сложных систем реального времени, используемых, например, в парадигме интернета вещей. Этот обзор включает оценку доступных решений и их пригодности для использования со следующим поколением сложных систем на кристалле с несколькими процессорными ядрами. Показано, что многие существующие решения не позволяют разработчикам легко воспользоваться преимуществами сложных функций, интегрированных в SoC следующего поколения. Обобщены и обсуждены основные функции поддержки отладки для многоядерных SoC. Даны рекомендации для разработчиков SoC и для будущего направления исследований в этой области в целях обеспечения более подходящей основы для новых инструментальных средств разработки. Такие средства крайне необходимы для всех встраиваемых систем жесткого реального времени и имеют первостепенное значение для минимизации сложности их разработки*

***Ключевые слова:*** *системы на кристалле, аппаратная отладка, SoC, цепочки сканирования, JTAG, трассировка*

## Введение

От современных систем и устройств, реализующих парадигму интернета вещей (*Internet of Things,* IoT), все чаще требуется достаточно высокая производительность в режиме реального времени при сохранении низкого энергопотребления. Эти требования приводят к созданию многоядерных SoC-решений с поддержкой широкого спектра периферийных устройств и коммуникационных протоколов. Системы дополнительно ограничены требованиями к работе в суровых условиях, например, в моторном отсеке автомобиля или на радиолокационной вышке. Разработка встраиваемых систем жесткого реального времени, в которых неуспевшая завершиться в срок задача может привести к физическому повреждению устройства, является сложным процессом.

Следует отметить, что основными решениями разработки SoC (в том числе и отладки в процессе разработки) являются моделирование и верификация. Для этого используют средства, подобные FireSim [1], способные выполнять детальное моделирование на неограниченных по производительности облачных ресурсах с привлечением для адекватности моделирования аппаратных ресурсов. Тем не менее создать систему, полностью свободную от ошибок, невозможно.

В настоящей работе речь пойдет о средствах и методах обнаружения ошибок, проявляющихся непосредственно в ходе эксплуатации систем. Среди них выделяют традиционные инструментальные средства, такие как интерактивные отладчики и профилировщики. Они крайне необходимы для разработки надежных встраиваемых систем. Современные технологии позволяют ин-

* Публикация выполнена в рамках государственного задания по проведению фундаментальных исследований по теме "Исследование и реализация программной платформы для перспективных многоядерных процессоров" (FNEF-2022-002).

тегрировать всю систему на одном кремниевом чипе, известном как система на кристалле, что приводит к перемещению существующих внешних интерфейсов, широко используемых в целях разработки, на чип. Традиционно связь внутри встраиваемой системы осуществляется с использованием внешней системной шины процессора, которая реализована в виде дорожек на печатной плате. Печатная плата должна поддерживать соответствующие интерфейсы инструментов разработки, требующих физического подключения, например, логических анализаторов и осциллографов. Размещение ранее внешних интерфейсов на кристалле оставляет все меньше вариантов для внешних инструментов анализа и, фактически, делает разработчиков "слепыми" к внутреннему состоянию SoC. Без надежного и последовательного представления состояния встраиваемой системы обнаружение дефектов или ошибок в ней может стать очень трудной или даже не имеющей решения задачей. Основным решением задачи отсутствия интерфейсов является предоставление доступа к внутренним узлам системы извне через существующие интерфейсы. Существуют различные подходы к достижению требуемого при этом уровня наглядности, которые в настоящей работе изложены в общих чертах.

Поддержка процесса отладки является жизненно важной частью этапа разработки встраиваемой системы. Причина в том, что независимо от того, насколько тщательно проводится работа на этапах проектирования или тестирования, возникают ошибки, которые можно обнаружить только когда система работает. Источники таких ошибок включают неправильно определенные (сформулированные) или неправильно понятые спецификации, изменения оборудования на стадии производства и наиболее распространенный источник — человеческий фактор.

По оценкам Национального Института Стандартов и Технологий (NIST), ошибки и сбои обходятся только экономике США примерно в 59,9 млрд долл. США в год [2]. Улучшение качества готовой системы за счет более тщательного тестирования и верификации может снизить эти затраты на одну треть. Отмечается, что 77 % электронных сбоев в автомобилях были связаны с программным обеспечением [3]. Есть все основания предполагать, что число проблемных вопросов отладки встраиваемых систем и систем на кристалле будет только возрастать по мере развития технологий.

Как правило, в основе SoC-устройств лежат системы жесткого реального времени. Такие системы являются одними из самых сложных в реализации в условиях дефицита аппаратных и программных ресурсов. Современные подходы к проектированию систем жесткого реального времени основаны на детерминированном времени выполнения задач. Даже небольшое изменение времени может отрицательно сказаться на механизмах, которыми эти системы управляют. Например, если контроллер двигателя пропустит крайний срок запуска или запустит импульс слишком рано, двигатель может быть выведен из строя. Недетерминированность в системах жесткого реального времени настолько проблематична, что многие системные интеграторы даже не используют кеш-память, повышающую производительность, и нередко отказываются от преимуществ использования операционных систем реального времени, предпочитая реализовывать задачи непосредственно для выполнения на аппаратуре [4].

## Существующие стратегии аппаратной отладки

Среди основных требований к эффективной поддержке процесса отладки аппаратуры можно выделить следующие:

- поддержка отладки не должна существенно изменять поведение устройства;
- наличие необходимой инфраструктуры для внешнего наблюдения за внутренним состоянием системы и другими критическими узлами;
- обеспечение внешнего доступа для управления состоянием системы и ресурсами, включая сложные периферийные устройства;
- средства поддержки аппаратной отладки не должны слишком усложнять SoC-устройство с точки зрения дополнительных контактов или площади чипа.

Повторное использование дизайна имеет решающее значение при разработке систем на кристалле для обеспечения своевременного выхода на рынок, поэтому поддержка отладки должна быть легко адаптируемой к различным системным архитектурам.

Существуют также различные методы отладки, которые обычно поддерживаются инфраструктурой отладки. Независимо от выбранной стратегии поддержки отладки базовая архитектура обычно содержит ряд основных частей. В следующих разделах дан обзор этих частей.

Одним из методов является посмертная отладка, которая останавливает SoC в ответ на такое событие, как, например, доступ по определенному адресу. В большинстве систем ошибкам требуется

время, чтобы проявить себя, они могут оставлять "улики", когда система остановлена, что упрощает поиск причины. К сожалению, некоторая часть этих улик исчезает к моменту остановки SoC, что затрудняет поиск ошибок и требует больших затрат на разработку. Посмертная отладка — это традиционный способ отладки встраиваемых систем с одним процессором [5], однако он имеет серьезные ограничения. Альтернативный подход заключается в наблюдении за системой во время ее работы.

Теперь рассмотрим четыре типа инфраструктуры отладки.

***Программная поддержка отладки.*** Ресурсы поддержки отладки могут быть программными или аппаратными. Программная поддержка отладки реализуется с помощью процедур мониторинга [6, 7], в которых программное обеспечение компилируется или собирается с дополнительным программным кодом, осуществляющим отладочные действия при наступлении некоторого события, например, обращения по некоторому адресу или выполнению определенной команды процессора. Программное обеспечение может быть независимым от платформы и часто поддерживается в процессорах с помощью инструкций отладки или специальным режимом отладки [8]. Примером программного инструментария является использование так называемых print-процедур в программном коде для отображения или протоколирования некоторых переменных состояния системы. Помимо средств мониторинга следует упомянуть средства профилирования с поддержкой со стороны аппаратуры, например, интерфейс доступа к специальным регистрам-счетчикам процессора *Performance Application Programming Interface*, PAPI [9].

Поддержка процесса отладки на программном уровне почти всегда сопровождается средствами базовой аппаратной отладки [8], включая как традиционное стендовое оборудование, так и базовую поддержку отладки на кристалле. Стендовое оборудование включает в себя логические анализаторы и осциллографы. Базовая поддержка отладки на кристалле обеспечивает минимальный интерфейс для управления SoC. Поддержка отладочного программного обеспечения может быть активна в штатном режиме, однако при этом она может конфликтовать с системными задачами, существенно влияя на поведение системы. Следует учесть, что активное вмешательство в процесс работы SoC не всегда оправдано, особенно для систем реального времени. Остановка и последующее возобновление работы системы обычно изменяют взаимосвязь системы и окружения (периферийных устройств).

Навязчивый характер программных средств отладки делает их малопригодными для систем жесткого реального времени. Любые отладочные механизмы, используемые во время разработки, как правило, будут удалены или остановлены в конечном "боевом" варианте системы, изменив тем самым общее поведение [10]. Вторым недостатком этой стратегии является то обстоятельство, что, фактически, она поддерживает только "посмертную" отладку, которая предоставляет данные только после того, как ошибки проявляются, а не тогда, когда они действительно происходят. Тем не менее такое программное обеспечение очень полезно в некоторых приложениях и его гораздо безопаснее использовать в системах с мягкими сроками выполнения, например, в системах, не контролирующих критически важные аппаратные узлы, особенно, если система использует не все доступные ресурсы.

***Отладка с использованием интерфейсов тестирования устройств.*** В настоящее время обычной практикой является использование технологии *design-for-test* (DFT) при разработке интегральных схем для поддержки производственного тестирования и отладки [6, 11]. Обычно она строится на механизме цепочек сканирования (*scan chains*) в соответствии со стандартом, обсуждаемым далее. В этот механизм включаются многие критически важные данные и способы управления SoC, что позволяет считывать состояние триггеров системы. В сочетании со средствами программной поддержки процесса отладки такой подход позволяет обеспечить простой инструментарий наблюдения и контроля. Основное преимущество решения на основе цепочек сканирования заключается в том, что оно использует непосредственно инфраструктуру DFT, поэтому не увеличивает производственные затраты.

Обычные цепочки сканирования используют один путь сканирования для уменьшения накладных расходов на маршрутизацию, но это ограничивает производительность. Еще одним недостатком является тот факт, что они обычно доступны только в режиме тестирования, поскольку цепочки сканирования мультиплексируются на контакты устройства, используемые для других целей в функциональном режиме. Использование выделенного тестового интерфейса позволяет преодолеть эту сложность. На практике производственные тесты преодолевают ограничения производительности за счет мультиплексирования нескольких цепочек сканирования на запасные

функциональные контакты [12]. Однако в функциональном режиме запасных контактов нет, поэтому такой подход не годится для поддержки отладки. Одним из решений этой задачи является введение дополнительного режима отладки, в котором цепочки сканирования объединяются вместе [13] или мультиплексируются [14]. Тогда они становятся доступны через выделенный интерфейс с одной линией сканирования.

Работающие цепочки сканирования приводят к перемещению данных, что потенциально изменяет выходные данные. Использование цепочек сканирования в неактивном состоянии позволяет избежать повреждения состояния системы, но делает невозможной ее отладку. В работе [15] приведены некоторые подходы, позволяющие проводить отладку SoC вместе с применением цепочек сканирования.

***Устройства встроенной эмуляции.*** Встроенный эмулятор (*In Circuit Emulator*, ICE) — это специально изготовленный прототип, заменяющий обычную производственную SoC. Такой эмулятор помогает преодолеть ограничения отладки на основе тестовой инфраструктуры или базовой аппаратной и программной инструментальной отладки, предоставляя дополнительные возможности подключения отладочных инструментов. Множество дополнительных отладочных контактов, расположенных на расширенной площади платы, не используются системой в функциональном режиме, вместо этого они подключены к внутренним выводам для обеспечения возможности наблюдения за поведением системы с помощью, например, логического анализатора. Следует отметить, что отладочные данные в этом случае потенциально доступны во время нормальной работы системы, поэтому, в принципе, использование дополнительных контактов не влияет на поведение SoC.

Недостатком такого подхода становится увеличенный размер микросхемы и наличие дополнительных соединений внутри нее, зачастую с использованием длинных соединительных проводов. Это обстоятельство отрицательно сказывается на надежности микросхемы, что может привести к полной ее непригодности для эксплуатации в суровых условиях, таких как отсек двигателя или коробки передач. Кроме того, подключение дополнительных соединительных проводов нагружает внутреннюю коммуникацию, снижая производительность и изменяя поведение прототипа по сравнению с реальным устройством, внося таким образом, дополнительные потенциальные ошибки.

***Поддержка отладки на основе эмуляции на кристалле.*** Более распространенной альтернативой встроенным ICE является эмуляция на кристалле (*on-chip emulation*), расширяющая возможности отладки обычной SoC за счет создания на самом кристалле дополнительных инструментальных средств сбора данных [16]. Такие средства обычно включают в себя логику управления точками останова, позволяющими останавливать систему при достижении определенного адреса в сегменте кода или доступе к определенной ячейке памяти. Аппаратные ресурсы отладки работают параллельно с обычным режимом функционирования SoC, поэтому поведение системы практически не меняется. Согласованное поведение системы делает стратегии отладки на основе самоэмуляции подходящими для большинства приложений, включая разработку сложных систем реального времени.

Технология эмуляции на кристалле позволяет использовать такой инструментарий, как трассировка, которая предоставляет разработчикам достаточную информацию об активности системы во время выполнения для последующего углубленного анализа.

Еще одним преимуществом трассировки является ее способность фиксировать события, приводящие к возникновению проблемы, а не только ее последствия. Поддержание правильного временного порядка событий также имеет важное значение для предоставления разработчику достоверной информации.

Трассировка на кристалле значительно отличается от встроенных ICE, которые выдают только необработанные сигналы. Встроенная трассировка допускает первичную обработку данных, например, их сжатие, необходимое для наблюдения за несколькими высокоскоростными встроенными процессорами и доступом к данным.

По мере развития интегральных схем развивалась и поддержка отладки. Каждая из четырех рассмотренных технологий имеет свои преимущества и недостатки. Современная тенденция отладки заключается в том, чтобы в значительной степени фокусироваться на отслеживании труднообнаруживаемых спорадических ошибок, поскольку их иногда невозможно обнаружить с помощью методов традиционной посмертной отладки. Профилирование и оптимизация систем реального времени также в большой степени зависят от наличия средств аппаратной трассировки.

## Управление SoC с несколькими процессорными ядрами

Сложные многоядерные SoC-устройства с несколькими процессорами и периферийными устройствами ставят новые проблемные вопросы

разработчикам SoC. Современная SoC — это не просто интеграция нескольких схем на печатной плате, а новый подход к проектированию, требующий эволюции методов проектирования.

***Интерфейсы управления запуском и доступом к памяти.*** Традиционным интерфейсом разработки SoC является так называемый интерфейс управления запуском (*run control*). Он используется для управления работой микроконтроллера или одноядерного процессора SoC. В частности, он позволяет разработчику запускать и останавливать такие устройства, как процессор.

Некоторые устройства SoC используют свой собственный проприетарный интерфейс [17], однако широко распространенным решением является использование существующего порта тестирования граничного сканирования (JTAG) [18]. Применение в целях отладки стандартизированного интерфейса тестирования не требует дополнительных выводов на плате, что дает этому подходу потенциал для внедрения на разных платформах. Системы с несколькими процессорами на печатной плате ранее включали несколько интерфейсов отладки, часто основанных на стандарте IEEE 1149.1, который первоначально предназначался для структурного тестирования цифровых печатных плат. Ранние SoC-устройства с несколькими процессорными ядрами были сконструированы путем интеграции большинства некогда отдельных компонентов на одной системной печатной плате. Многие из ядер имеют собственные встроенные контроллеры TAP (*Test Access Port*), что делает актуальным вопрос о том, как управлять множеством контроллеров TAP, сохраняя при этом соответствие стандарту IEEE 1149.1. В работе [15] предложены несколько вариантов решения этого вопроса, включая популярный вариант включения всех контроллеров TAP в одну цепочку сканирования.

Однако предложенный в работе [15] так называемый агент-ориентированный подход видится авторам более удачным решением. По сути, этот подход является следующим эволюционным шагом в развитии систем на кристалле, обеспечивая внедрение системно-ориентированной парадигмы, в которой каждая SoC имеет один контроллер TAP для связи с внешними инструментами отладки и тестирования [19]. Следует отметить, что устаревшие процессорные ядра не могут быть изменены, поэтому придется их поддерживать, пока они не будут выведены из эксплуатации. Контроллер TAP на уровне кристалла целиком, который взаимодействует с интерфейсами отладки каждого ядра, обеспечивает более масштабируемую архитектуру.

Существует реализация интерфейса, называемая JTAG+, сочетающая в себе аспекты как центральной, так и распределенной архитектур [16]. В ней используется один центральный контроллер TAP и цепочка сканирования для связи со специальным отладочным процессором (агентом отладки), который, в свою очередь, использует специальную системную шину для доступа к памяти и внутреннему состоянию остальных процессоров SoC. Это существенно отличается от таких решений, как EJTAG [20], где отладочный сопроцессор привязан к каждому процессорному ядру и доступен с помощью цепочек сканирования. Устаревшие ядра поддерживаются путем подключения цепочек сканирования к центральному контроллеру TAP, который обеспечивает выбор между встроенными контроллерами TAP и агентом отладки.

***Контроль хода выполнения SoC-устройств с несколькими ядрами и таймерами.*** В большинстве современных SoC-устройств точки останова поддерживаются аппаратными триггерами. В системах с трассировкой триггеры также могут использоваться для запуска и остановки сбора трассы, а также фильтрации событий для сохранения пропускной способности и объема памяти. Этот процесс фильтрации известен как квалифицирование трассы, поскольку выбираются только наиболее полезные для последующего анализа события.

Большинство сложных SoC-устройств сейчас имеют несколько системных таймеров, что усложняет синхронизацию событий, произошедших на разных ядрах. Это обстоятельство также влияет на отладку, поскольку усложняет остановку работающей системы целиком. При использовании одного системного таймера вся SoC может быть остановлена одновременно, но с разными или несинхронизированными таймерами существует вероятность рассинхронизации данных [21]. Хотя рассинхронизация данных может быть обнаружена, вопрос с перезапуском SoC остается. По-прежнему одним из главных отладочных действий остается полный останов и последующий запуск всей системы на кристалле. Чтобы избежать изменения поведения системы, все ее компоненты должны быть остановлены в один и тот же момент времени. Одним из решений является построение распределенного коммутатора для всех отладочных шин [22]. Преимуществом такого решения является тот факт, что компоненты SoC подключаются к общему блоку, действующему как центральный узел для детерминированной маршрутизации сигналов прерывания с минимальной задержкой. Использование коммутатора позволяет разработчику настроить, какие ядра будут реагировать на сигналы прерывания.

## Поддержка трассировки

Рассмотренные до сих пор функции поддержки процесса отладки в основном связаны с контролем и наблюдением за состоянием системы в отдельные моменты времени. Чтобы найти сложные ошибки, разработчикам необходимо просмотреть срезы состояния системы за некоторое время до проявления ошибки. Чтобы предоставить доступ к этой информации, SoC может получать и обрабатывать данные о своем состоянии в режиме реального времени. Затем полученная трасса становится доступной для внешних средств разработки либо через порт отладки/трассировки [7, 20, 22—25], либо путем сохранения во встроенной памяти, откуда ее можно считывать в автономном режиме с помощью интерфейса управления запуском [20, 24]. Трассировка — это не просто полезный инструментарий для отладки одноядерных систем. Он также крайне необходим для отладки взаимодействия между несколькими процессорными ядрами и активными периферийными устройствами.

Стандартные инструментальные средства сбора данных о трассе просто регистрируют поток управления процессорного ядра [25], протоколируя тип выполняемых инструкций в порядке их появления. Существуют также инструментальные механизмы регистрации событий доступа к данным [23], которые помогают найти ошибки, связанные с общими переменными, блокировками и другими взаимодействиями, где важен порядок событий. Отслеживание данных всех активных устройств также очень важно, поскольку не все взаимодействия связаны с процессором [10]. Базовая трасса программы может содержать только информацию, достаточную для восстановления потока управления в автономном режиме. Она может быть дополнена подробными сведениями о, например, запущенной задаче или возникшем прерывании. Трассы критически важных систем, применяемых, например, в космических аппаратах, могут включать гораздо больше деталей [26].

Типичная трасса потока данных содержит такую информацию, как начальный адрес данных, их размер и значения, а также флаг доступа — чтение или модификация. Периферийные устройства могут также предоставлять дополнительную информацию о состоянии, такую как пропускная способность интерфейса или статистика возникновения ошибок соединения.

Информация о состоянии, не требующая анализа в реальном времени, обычно считывается из регистров, отображенных в памяти, с помощью интерфейса управления запуском.

***Сжатие трассы.*** Некоторые инструментальные средства сбора данных о трассе при протоколировании данных от потока управления генерируют простое в реализации сжатое сообщение фиксированной ширины для каждого процессорного такта [8, 25]. Альтернативная стратегия, определенная стандартом NEXUS [23] и используемая некоторыми производителями SoC [7], заключается в том, чтобы формировать сообщения различной длины и отправлять их только при необходимости. Уменьшение размера сообщений в стандарте NEXUS достигается за счет того, что зачастую программы, выполняемые на процессоре, обращаются в память на коротком диапазоне адресов (например, обработка массивов). В этом случае возможно хранить и передавать только часть адреса, а не весь адрес целиком (так называемое разностное сжатие, *differential compression*). Некоторые производители пользуются универсальными алгоритмами сжатия данных, такими как коды Хаффмана [27].

В стандарте NEXUS описывается алгоритм сжатия данных о трассе, в котором сообщения отправляются только для событий, требующих синхронизации, таких как инструкции ветвления или trap-инструкции. Вместо отдельных сообщений для последовательных инструкций подсчитывается число последовательных инструкций, выполненных с момента формирования предыдущего сообщения, и это число включается в следующее сообщение. Фактически речь идет о протоколировании не отдельных инструкций, а целых линейных участков кода. Такой подход позволяет значительно уменьшить размер трассы, но при условии, что у инструментального средства анализа данных о трассе имеется доступ к исходному коду, используя который можно воспроизвести поведение программы.

Наиболее эффективным методом уменьшения объема трассы является возможность управления трассировкой на уровне отдельных аппаратных модулей SoC, а также фильтрация событий по разным условиям. К таким относятся, например, события только чтения или только записи данных, события выполнения только инструкций сопроцессора плавающей арифметики и т. п. [28].

***Управление трассировкой.*** Кроме способов генерации трассы также важны методы управления сбором трассы и ее хранения. Сбором данных о трассе можно управлять вручную через внешний интерфейс. Однако управление с помощью определенных аппаратно-программных триггеров более эффективно, поскольку позволяет разработчикам получить данные о системе в момент, непосредственно предшествующий триггерному событию ("до"-данные), или сразу за ним последу-

ющий ("после"-данные) [28]. Сбор "после"-данных полезен в ситуации, когда разработчик подозревает о наличии некорректного поведения системы, начиная с определенного события, в то время как сбор "до"-данных нужен, когда разработчик не уверен в причине ошибки и хочет увидеть изменение поведения системы вплоть до проявления ошибки.

Самый простой подход к трассировке системы с несколькими ядрами заключается в том, что средство управления трассировкой переключается между каждым ядром — источником трассы [22]. Однако такой подход не ориентирован на систему целиком и имеет ограниченное применение там, где ядра взаимодействуют между собой.

Настоящая системно-ориентированная комбинированная трассировка должна обеспечивать согласованное представление о взаимодействиях внутри системы. Алгоритм объединения сообщений от каждого источника является важным фактором в увеличении эффективности трассировки. При наличии разных системных таймеров необходимо уметь упорядочивать сообщения, используя логическое время. В этом случае разработчик получит достоверную картину того, что происходит внутри SoC в целом. Полученная такая образом трасса позволяет разработчику обнаруживать ошибки, связанные с параллелизмом, включая доступ к общим переменным. Одним из решений, способных объединить трассы из нескольких источников в правильном временном порядке, является сеть на кристалле (*Network-on-Chip*, NoC [29]).

В устройствах SoC с разнородными процессорными ядрами часто нет другого выбора, кроме как пытаться объединить данные от несколько отдельных систем трассировки, поскольку каждое семейство процессоров имеет собственные реализации отладочных средств и интерфейсов. Введение общего стандартизированного интерфейса отладки решило бы задачу системной интеграции.

## Рекомендации по поддержке отладки

Для отладки сложных систем на кристалле разработчикам требуются инструментальные средства, учитывающие наличие нескольких процессорных ядер и активных периферийных устройств, которые являются ключевыми компонентами высокопроизводительных встраиваемых систем. Различные архитектуры ядер, наличие нескольких системных таймеров, требования к работе в реальном времени и повышенная сложность других компонентов систем на кристалле показали ограничения традиционных методов посмертной отладки. Применение инфраструктуры отладки и тестирования (JTAG, JTAG+, EJTAG) для отладки всей системы на кристалле не является жизнеспособным методом. В основном это связано с тем обстоятельством, что тестовая инфраструктура сосредоточена почти исключительно на деятельности, связанной с тестированием. Поэтому исследование и разработка новых методов, учитывающих все аспекты отладки, принесло бы пользу SoC. Отметим, что это решение должно заменить устоявшиеся методологии тестирования.

Механизм аппаратных точек останова, по сути, остается незаменимым средством отладки. Дополнение этого механизма функциональными возможностями трассировки (вместо создания отдельных аппаратных средств трассировки) поможет снизить накладные расходы на поддержание средств отладки.

Программные методы и средства отладки хорошо изучены и будут оставаться ценным инструментарием. Однако для повышения эффективности неразумно при проектировании SoC полагаться только на программное обеспечение.

Для систем реального времени трассировка является минимальным требованием и должна быть усилена наличием механизмов фильтрации и аппаратными триггерами. Многоядерным SoC требуется инфраструктура для объединения механизмов сбора трассы каждого компонента, с сохранением при этом истинного временного порядка их сообщений с идентификацией источника. Пока существуют только базовые решения, следовательно, необходимо разрабатывать и исследовать новые методологии для поддержки будущих поколений более сложных систем на кристалле.

Одной из таких методологий, по мнению авторов, может являться концепция контролируемого выполнения [30]. В этой концепции все инструментальные средства отладки тесно интегрированы. Кроме того, сама система на кристалле еще на этапе проектирования получает именно те средства отладки, которые направлены на локализацию и устранение типичных ошибок для той целевой области, под которую эта система разрабатывается. Разработчики аппаратных компонентов и программного обеспечения должны понимать, что им необходимо тесно сотрудничать с точки зрения определения задач и для достижения практических результатов.

При современном подходе к разработке SoC, ориентированном на IP-блоки, важно, чтобы поддержка процессов разработки и отладки стала самостоятельным блоком многократного использования. Такой блок должен быть отделен от платформ и процессоров, он должен обладать спецификой для конкретного поставщика, иметь стандартизированные интерфейсы, подобными тем, которые определены в работе [31].

## Заключение

Системы на кристалле эволюционировали до очень сложных устройств, но при этом поддержка средств их разработки и отладки фактически осталась на прежнем уровне. Возрастающая сложность архитектуры с одной стороны и повышенные требования к высокому уровню качества и надежности с другой стороны сделали поддержку разработки решающим фактором для успешного создания новых современных систем на кристалле. С помощью "правильных" инструментальных средств можно решить вопросы соответствия новым повышенным требованиям современных сложных систем на кристалле. Для продвижения интеграции системы важно, чтобы поддержка разработки SoC перестала фокусироваться исключительно на недорогих и менее эффективных архитектурах. Будущее отладки, несомненно, в системно-ориентированным решении. Как показано в настоящей работе, здесь был достигнут определенный прогресс, но многие разработчики SoC не спешат внедрять новые методы и инструментальные средства отладки, в которых остро нуждается конечный пользователь.

В настоящей работе выделены области, в которых есть возможность для дальнейших исследований. Очевидно, что у каждой системы на кристалле есть свое собственное предназначение и следующие из этого требования. Соответственно, по мере развития технологий необходимо и совершенствование средств отладки таких систем.

### Список литературы


1. **FireSim.** URL: https://fires.im
2. **US** Department of Commerce. The economic impacts of inadequate infrastructure for software testing, Technical Report, RTI-7007.011US, National Institute of Standards and Technology (US), 2002. 309 p. URL: https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf
3. **Mayer A., McDonald-Maier K. D.** Debug support, calibration and emulation for multiple processor and powertrain control SoCs [automotive applications] // Design, Automation and Test in Europe. Vol. 3. Munich (DE). 7—11 March 2005. P. 148—152.
4. **Scottow R. G., McDonald-Maier K. D.** Measuring determinism in real-time embedded systems using cached processors//Proceedings of the 2005 International Conference on Embedded Systems and Applications, ESA'05. Las Vegas, 7—11 March 2005. P. 38—44.
5. **Huang I.-J., Lu T.-A.** ICEBERG: an embedded in-circuit emulator synthesizer for microcontrollers // Proceedings of the Design Automation Conference. 1999. P. 580—585. DOI: 10.1109/DAC.1999.94.
6. **Zorian Y., Jan Marinissen E., Dey S.** Testing embedded-core-based system chips // Computer. 1999. Vol. 32, No. 6. P. 52—60. DOI: 10.1109/2.769444.
7. **Motorola Inc.** MPC565/MPC566 user's manual, 1387 p. URL: https://seniord.ece.iastate.edu/projects/archive/may0715/resources/MPC565_MPC566RM.pdf
8. **Infineon** Technologies AG. Tricore 1 architecture manual, ver. 1.3.8, January 2008, 280 p. URL: https://www.infineon.com/dgdl/tc_v131_corearchitecture_v__138.pdf?fileId=db3a304412b407950112b409c4500359
9. **Performance** Application Programming Interface, PAPI. URL: https://cvw.cac.cornell.edu/Profiling/papi
10. **Mayer A., Siebert H., Kolof A., el Baradie S.** Debug support for complex system-on-chips // Proceedings of the Embedded Systems Conference, April 2003. P. 1—16.
11. **Golshan F.** Test and on-line debug capabilities of IEEE Std 1149.1 in UltraSPARCTM-III microprocessor // Proceedings of the International Test Conference, October 2000. P. 141—150. DOI: 10.1109/TEST.2000.894201.
12. **Vermeulen B., Goel S. K.** Design for debug: catching design errors in digital chips // IEEE Design & Test of Computers. 2002. Vol. 19, No. 3. P. 35—43.
13. **Van Rootselaar G. J., Vermeulen B.** Silicon debug: scan chains alone are not enough // Proceedings of the International Test Conference (IEEE Cat. No. 99CH37034). 1999. P. 892—902. DOI: 10.1109/TEST.1999.805821.
14. **Jung D.-J., Kwak S.-H., Lee M.-K.** Reusable embedded debugger for 32 bit RISC processor using the JTAG boundary scan architecture // Proceedings of the IEEE Asia-Pacific Conference on ASIC, 2002. P. 209—212. DOI: 10.1109/APASIC.2002.1031569.
15. **Hopkins A. B. T., McDonald-Maier K. D.** Debug support for complex systems on-chip: a review // Computers and Digital Techniques. 2006. No. 4. P. 197—207. DOI: 10.1049/ip-cdt:20050194.
16. **Maier K. D.** On-chip debug support for embedded systems-on-chip // Proceedings of the International Symposium on Circuits and Systems, Bangkok, Thailand. 25—28 May 2003. P. 565—568. DOI: 10.1109/ISCAS.2003.1206375.
17. **Melear C.** Using background modes for testing, debugging and emulation of microcontrollers // Proceedings of the WESCON/97 Conference, 1997. P. 90—97. DOI: 10.1109/WESCON.1997.632324.
18. **IEEE** JTAG 1149.1-2013 Std. IEEE standard test access port and boundary-scan architecture, IEEE Computer Society, 2013. 444 p.
19. **Oakland S. F.** Position statement: TAPs all over my chips // Proceedings of the International Test Conference. 7—10 October 2002. P. 1192—1193. DOI: 10.1109/TEST.2002.1041899.
20. **MIPS** Technologies. EJTAG trace control block specification, MD00148 Rev. 1.04, 2002. 66 p. URL: http://www.t-es-t.hu/download/mips/md00148a.pdf
21. **Goel S. K., Vermeulen B.** Hierarchical data invalidation analysis for scan-based debug on multiple-clock system chips // Proceedings of the International Test Conference (ITC02). Baltimore. USA, 7—10 October 2002. P. 1103—1110. DOI: 10.1109/TEST.2002.1041867.
22. **Infineon** Technologies AG. Infineon TC1920 system units user's guide v1.3, October 2003. 78 p. URL: https://www.alldatasheet.com/datasheet-pdf/pdf/80124/INFINEON/TC1920.html
23. **IEEE-ISTO** 5001-2012 Std. Standard for a global embedded processor debug interface, Version 3.0. The Nexus 5001Forum, 2012. 190 p. URL: https://nexus5001.org/wp-content/uploads/2018/05/IEEE-ISTO-5001-2012-v3.0.1-Nexus-Standard.pdf
24. **Zeinolabedin S., Partzsch J., Mayr C.** Real-time Hardware Implementation of ARM CoreSight Trace Decoder // IEEE Design & Test. 2021. Vol. 38, No. 1. P. 69—77. DOI: 10.1109/MDAT.2020.3002145.
25. **Renesas** Technology Corp. Hitachi SuperH RISC engine SH7144 Series hardware manual, Rev. 2.0, 2002. 773 p. URL: https://datasheet.octopart.com/HD64F7145F50V-Renesas-datasheet-154546106.pdf
26. **Gaisler Research.** GRLib, 2008. 77 p. URL: https://opencores.org/websvn/filedetails?repname=mips_enhanced&path=%2Fmips_enhanced%2Ftrunk%2Fgrlib-gpl-1.0.19-b3188%2Fdoc%2Fgrlib.pdf
27. **Huffman D. A.** A method for the construction of minimum redundancy codes // Proceedings of the IRE. 1952. Vol. 40, No. 9. P. 1098—1101. DOI: 10.1109/JRPROC.1952.273898.
28. **ARM.** Embedded trace macrocell architecture specification, 2011. URL: https://developer.arm.com/documentation/ihi0014/q/
29. **Benini L., De Micheli G.** Networks on chips: a new SoC paradigm // Computer. 2002. Vol. 35, No. 1. P. 70—78. DOI: 10.1109/2.976921.
30. **Бетелин В. Б., Галатенко В. А., Костюхин К. А.** Контролируемое выполнение приложений на многопроцессорных платформах // Информационные технологии. 2015. Том 21, № 10. С. 723—728.
31. **Hopkins A. B. T., McDonald-Maier K. D.** Debug support strategy for systems-on-chips with multiple processor cores // IEEE Transactions on Computers. 2006. Vol. 55, No. 2. P. 174—184. DOI: 10.1109/TC.2006.22.

# Hardware Debugging: an Overview of Modern Approaches

**V. A. Galatenko,** galat@niisi.ras.ru, **K. A. Kostyukhin**, kost@niisi.ras.ru,
Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences", Moscow, 117218, Russian Federation

*Corresponding author:*

**Konstantin A. Kostyukhin**, Senior Researcher, Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences", Moscow, 117218, Russian Federation
E-maill: kost@niisi.ras.ru

*Received on July 12, 2022*
*Accepted on July 27, 2022*

*The ubiquity of complex devices built on systems on chip (SoC) with multiple processor cores poses new challenges for developers of embedded systems. New development tools specifically designed for complex systems on chip can help solve them, but these tools are usually limited by the functionality of debugging support tools. High-quality debugging support with advanced features is necessary to take full advantage of complex SoC devices while reducing development time. The article discusses various mechanisms and ways to implement support for debugging systems on chip designed for complex real-time systems used, for example, in the Internet of Things (IoT) paradigm. This review includes an assessment of the available solutions and their suitability for use with the next generation of complex systems on chip with multiple processor cores. It is shown that many existing solutions do not allow developers to easily take advantage of complex functions integrated into the next-generation SoC. The basic debugging support functions for multicore SoCs are summarized and discussed. Recommendations are given for SoC developers and for the future direction of research in this area in order to provide a more suitable basis for new development tools. Such tools are extremely necessary for all embedded hard real-time systems and are of high importance for minimizing the complexity of their development.*

*Modern systems and devices implementing the Internet of Things paradigm are increasingly required to have sufficiently high real-time performance while maintaining low power consumption. These requirements lead to the creation of multicore SoC solutions with support for a wide range of peripheral devices and communication protocols. The systems are limited by the requirements for working in harsh conditions, such as, for example, in the engine compartment of a car or on a radar tower. The development of embedded hard real-time systems, in which a task that fails to be completed on time can lead to physical damage of the device, is a complex process.*

*Effective tools, such as interactive debuggers and profilers, are an integral part of solving these problems and are vital for developing reliable embedded systems. Modern technologies now allow the integration of the entire system on a single silicon chip, known as a system on a chip (SoC), which leads to the relocation of existing external interfaces, widely used for development purposes, to the chip. Traditionally, communication within an embedded system is carried out using an external processor system bus, which is implemented in the form of tracks on a printed circuit board. The printed circuit board must support the appropriate interfaces of development tools that require a physical connection, such as, for example, logic analyzers and oscilloscopes. Previously, placing external interfaces on a chip left fewer options for external analysis tools and, in fact, makes developers "blind" to the internal state of the SoC. Without a reliable and consistent representation of the state of the embedded system, the detection of defects or errors in the system may become difficult or even impossible. The main solution to the problem of the lack of external interfaces is to provide access to internal nodes from outside the system through existing interfaces. There are many different approaches to achieving the required visibility, and they are outlined in this review.*

**Keywords:** *system on a chip, hardware debugging, SoC, scan chains, JTAG, tracing*

*For citation:*

**Galatenko V. A., Kostyukhin K. A.** Hardware Debugging: an Overview of Modern Approaches, *Programmnaya Ingeneria*, 2022, vol. 13, no. 9, pp. 415—424.

DOI: 10.17587/prin.13.415-424

## References


1. **FireSim,** available at: https://fires.im
2. **US** Department of Commerce. The economic impacts of inadequate infrastructure for software testing, Technical Report, RTI-7007.011US, National Institute of Standards and Technology (US), 2002, 309 p, available at: https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf
3. **Mayer A., McDonald-Maier K. D.** Debug support, calibration and emulation for multiple processor and powertrain control SoCs [automotive applications], *Design, Automation and Test in Europe*, vol. 3, Munich (DE), 7—11 March 2005, pp. 148—152.
4. **Scottow R. G., McDonald-Maier K. D.** Measuring determinism in real-time embedded systems using cached processors, *Proceedings of the 2005 International Conference on Embedded Systems and Applications, ESA'05*, Las Vegas, 7—11 March 2005, pp. 38—44.
5. **Huang I.-J., Lu T.-A.** ICEBERG: an embedded in-circuit emulator synthesizer for microcontrollers, *Proceedings of the Design Automation Conference*, 1999, pp. 580—585, DOI: 10.1109/DAC.1999.94.
6. **Zorian Y., Jan Marinissen E., Dey S.** Testing embedded-core-based system chips, *Computer*, 1999, vol. 32, no. 6, pp. 52—60. DOI: 10.1109/2.769444.
7. **Motorola Inc.** MPC565/MPC566 user's manual, 1387 p., available at: https://seniord.ece.iastate.edu/projects/archive/may0715/resources/MPC565_MPC566RM.pdf
8. **Infineon** Technologies AG. Tricore 1 architecture manual, ver. 1.3.8, January 2008, 280 p., available at: https://www.infineon.com/dgdl/tc_v131_corearchitecture_v__138.pdf?fileId=db3a304412b407950112b409c4500359
9. **Performance** Application Programming Interface, PAPI, available at: https://cvw.cac.cornell.edu/Profiling/papi
10. **Mayer A., Siebert H., Kolof A., el Baradie S.** Debug support for complex system-on-chips — Proceedings of the Embedded Systems Conference, April 2003, pp. 1—16.
11. **Golshan F.** Test and on-line debug capabilities of IEEE Std 1149.1 in UltraSPARCTM-III microprocessor, *Proceedings of the International Test Conference,* October 2000, pp. 141—150. DOI: 10.1109/TEST.2000.894201.
12. **Vermeulen B., Goel S. K.** Design for debug: catching design errors in digital chips, *IEEE Design & Test of Computers*, 2002, vol. 19, no. 3, pp. 35—43.
13. **Van Rootselaar G. J., Vermeulen B.** Silicon debug: scan chains alone are not enough, *Proceedings of the International Test Conference (IEEE Cat. No.99CH37034),* 1999, pp. 892—902. DOI: 10.1109/TEST.1999.805821.
14. **Jung D.-J., Kwak S.-H., Lee M.-K.** Reusable embedded debugger for 32 bit RISC processor using the JTAG boundary scan architecture, *Proceedings of the IEEE Asia-Pacific Conference on ASIC*, 2002, pp. 209—212. DOI: 10.1109/APASIC.2002.1031569.
15. **Hopkins A. B. T., McDonald-Maier K. D.** Debug support for complex systems on-chip: a review, *Computers and Digital Techniques*, 2006, no. 4, pp. 197—207. DOI: 10.1049/ip-cdt:20050194.
16. **Maier K. D.** On-chip debug support for embedded systems-on-chip, *Proceedings of the International Symposium on Circuits and Systems,* Bangkok, Thailand, 25—28 May 2003, pp. 565—568. DOI: 10.1109/ISCAS.2003.1206375.
17. **Melear C.** Using background modes for testing, debugging and emulation of microcontrollers, *Proceedings of the WESCON/97 Conference*, 1997, pp. 90—97. DOI: 10.1109/WESCON.1997.632324.
18. **IEEE** JTAG 1149.1-2013 Std**.** IEEE standard test access port and boundary-scan architecture, IEEE Computer Society, 2013, 444 p.
19. **Oakland S. F.** Position statement: TAPs all over my chips, *Proceedings of the International Test Conference*. 7—10 October 2002. P. 1192—1193. DOI: 10.1109/TEST.2002.1041899.
20. **MIPS** Technologies. EJTAG trace control block specification, MD00148 Rev. 1.04, 2002, 66 p., available at: http://www.t-es-t.hu/download/mips/md00148a.pdf
21. **Goel S. K., Vermeulen B.** Hierarchical data invalidation analysis for scan-based debug on multiple-clock system chips, *Proceedings of the International Test Conference (ITC02),* Baltimore, USA, 7—10 October 2002, pp. 1103—1110. DOI: 10.1109/TEST.2002.1041867.
22. **Infineon** Technologies AG. Infineon TC1920 system units user's guide v1.3, October 2003, 78 p., available at: https://www.alldatasheet.com/datasheet-pdf/pdf/80124/INFINEON/TC1920.html
23. **IEEE-ISTO** 5001-2012 Std**.** Standard for a global embedded processor debug interface, Version 3.0, The Nexus 5001Forum, 2012, 190 p., available at: https://nexus5001.org/wp-content/uploads/2018/05/IEEE-ISTO-5001-2012-v3.0.1-Nexus-Standard.pdf
24. **Zeinolabedin S., Partzsch J., Mayr C.** Real-time Hardware Implementation of ARM CoreSight Trace Decoder, *IEEE Design & Test,* 2021, vol. 38, no. 1, pp. 69—77. DOI: 10.1109/MDAT.2020.3002145.
25. **Renesas** Technology Corp**.** Hitachi SuperH RISC engine SH7144 Series hardware manual, Rev. 2.0, 2002, 773 p., available at: https://datasheet.octopart.com/HD64F7145F50V-Renesas-datasheet-154546106.pdf
26. **Gaisler** Research. GRLib, 2008, 77 p., available at: https://opencores.org/websvn/filedetails?repname=mips_enhanced&path=%2Fmips_enhanced%2Ftrunk%2Fgrlib-gpl-1.0.19-b3188 %2Fdoc%2Fgrlib.pdf
27. **Huffman D. A.** A method for the construction of minimum redundancy codes, *Proceedings of the IRE*, 1952, vol. 40, no. 9, pp. 1098—1101. DOI: 10.1109/JRPROC.1952.273898.
28. **ARM.** Embedded trace macrocell architecture specification, 2011, available at: https://developer.arm.com/documentation/ihi0014/q/
29. **Benini L., De Micheli G.** Networks on chips: a new SoC paradigm, *Computer*, 2002, vol. 35, no. 1, pp. 70—78. DOI: 10.1109/2.976921.
30. **Betelin V. B., Galatenko V. A., Kostyukhin K. A.** Multiprocessor applications controlled execution), *Informacionnye tehnologii,* 2015, vol. 21, no. 10, pp. 723—728 (in Russian).
31. **Hopkins A. B. T., McDonald-Maier K. D.** Debug support strategy for systems-on-chips with multiple processor cores, *IEEE Transactions on Computers*, 2006, vol. 55, no. 2, pp. 174—184. DOI: 10.1109/TC.2006.22.

УДК 004.67; 519.22; 330.46
DOI: 10.17587/prin.13.425-439

**Е. В. Орлова,** д-р техн. наук, проф., ekorl@mail.ru, Уфимский государственный авиационный технический университет

# Системный инжиниринг цифровых двойников организационно-технических систем с использованием методов интеллектуального анализа

*Рассмотрена проблема разработки цифровых двойников организационно-технических систем. Предложен методологический подход для организации процесса проектирования цифрового двойника организационно-технических систем, объединяющий этапы проектирования, методы и модели, а также обеспечивающий системный инжиниринг цифрового двойника. Разработана модель поддержки принятия решений для диагностики технического состояния объекта (технического устройства), основанная на методах ситуационного анализа и нечеткой логики, обеспечивающая синтез эффективных решений в различных ситуациях и сочетаниях разнородных факторов, характеризующих объект и его внешнюю среду.*

***Ключевые слова:*** *цифровой двойник, методы моделирования цифровых двойников, системное проектирование цифровых двойников, методы искусственного интеллекта*

## Введение

Драйвером инновационного развития высокотехнологичных предприятий в контексте четвертой промышленной революции становится технология "цифровой двойник" (ЦД) как виртуальный прототип реальных производственных процессов, изделий, готовых продуктов. Использование ЦД способствуют росту конкурентоспособности производимых изделий за счет повышения скорости вывода их на рынок. Применение ЦД обеспечивает обоснование решений за счет быстрой проверки изменений, вносимых в конструкцию изделия и его составных частей, в ходе цифровых испытаний.

В России впервые в мире разработана нормативно-техническая документация, регламентирующая процессы разработки и применения ЦД. В 2021 г. принят национальный стандарт "Компьютерные модели и моделирование. Цифровые двойники изделий. Общие положения"[1], определяющий общие положения построения и эксплуатации цифровых двойников изделий.

Однако организационно-методическое обеспечение процесса разработки и использования ЦД остается не до конца проработанным с точки зрения согласования задач по описанию объекта моделирования и управления. Для организации такой работы требуются применение системного подхода и проектирование объекта/процесса, учитывая разные аспекты — выполняемые функции, включая идентификацию и решение возникающих проблемных вопросов в процессе его функционирования, уровень сложности, назначение, этапы жизненного цикла и другие свойства и особенности.

Целью исследования, результаты выполнения которого представлены в статье, является разработка организационно-методического обеспечения проектирования ЦД организационно-технических систем, обеспечивающего системный синтез этапов проектирования, инструментария (методов и моделей) и результатов, направленных на ускоренный инжиниринг ЦД. Такое организационно-методическое обеспечение должно быть также применимо для разработки прототипа ЦД организационно-технических систем на этапе жизненного цикла "создание и разработка" с тем, чтобы исследовать возможные непредвиденные трудности и сбои, а также уменьшить последствия непредвиденных нежелательных состояний систем при их дальнейшей эксплуатации.

Для реализации цели решаются перечисленные далее задачи.

1. Обобщение подходов и методов моделирования и проектирования ЦД организационно-технических систем.

[1] ГОСТ Р 57700.37—2021. Компьютерные модели и моделирование. Цифровые двойники изделий. Общие положения.

2. Разработка методологии системного инжиниринга ЦД для организации процесса проектирования ЦД организационно-технических систем.

3. В рамках прототипа ЦД построение модели поддержки принятия решений для диагностики технического состояния устройства и принятия решений по устранению его неисправностей на основе методов искусственного интеллекта и нечеткой логики.

## 1. Методы моделирования и приложения цифровых двойников организационно-технических систем

### 1.1. Обобщение существующих подходов и методов моделирования цифровых двойников

Можно выделить три группы подходов и методов, используемых для проектирования ЦД:

- методы, основанные на математическом моделировании физических процессов (структурные модели объекта) (*Simulation-Based Digital Twin*) [1—3];
- методы, основанные на данных (*Data-based Digital Twin*) [4, 5];
- гибридные методы (*Hybrid Digital Twin*) [6, 7]

Особенности приведенных методов даны в табл. 1, где обобщены подходы к моделированию в рамках определенных характеристик.

Математическое моделирование обеспечивает отражение в модели физических свойств (принципов функционирования) объекта/процесса. Физическая модель обеспечивает компьютерное моделирование физических процессов, протекающих во времени, а также описание законов их функционирования и связей с внешней средой. Построение таких моделей на практике связано с применением методов математического программирования (исследования операций) [8], имитационного моделирования на основе разных его парадигм и подходов — системно-динамического, дискретно-событийного или агентного моделирования [9].

Математическое моделирование является ключевым компонентом ЦД. Для создания моделей, которые применяются при создании системных моделей, ЦД может использовать результаты детальных трехмерных численных расчетов, выполненных с применением междисциплинарных CAE-решателей. В зависимости от цели математические модели могут быть дескриптивными или оптимизационными. Целью дескриптивных моделей является установление законов изменения параметров модели. Оптимизационная модель обеспечивает поиск экстремума целевой функции при наличии ограничений с использованием численных методов. В зависимости от степени определенности исходной информации и учета временного фактора могут использоваться методы линейного, нелинейного, стохастического программирования, теоретико-игровые методы, методы нечеткой логики.

Имитационные модели как подкласс математических моделей делятся на статические и динамические; детерминированные и стохастические; дискретные и непрерывные. В непрерывных имитационных моделях переменные изменяются непрерывно, состояние моделируемой системы меняется как непрерывная функция времени, и, как правило, это изменение описывается системами дифференциальных уравнений. В дискретных имитационных моделях переменные изменяются дискретно в определенные моменты имитационного времени (наступления событий). Динамика дискретных моделей представляет собой процесс перехода от момента наступления очередного события к моменту наступления следующего события.

Моделирование, основанное на данных, включает в себя методы интеллектуального анализа данных, искусственного интеллекта, Big Data and Advanced Analytics. Каждый из этих методов предъявляет особые требования к необходимым вычислительным ресурсам. Например, методы интеллектуального анализа данных требуют хранилищ большого объема с высокой пропускной способностью для сбора и получения доступа к аналитическим данным, а также высокой масштабируемости вычислительной системы для их обработки; для применения методов машинного обучения требуются узлы с установленными графическими ускорителями.

Модели на основе интеллектуального анализа данных (*Data Mining*) применяют для обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия стратегически важных решений. Искусственный интеллект и машинное обучение эффективно используются в задачах прогнозирования в ЦД. Применение этих методов позволяет достичь уровня прогностической точности выше, чем на базе традиционных методов имитационного моделирования [10].

Использование "больших данных" имеет свои ограничения, связанные с неполными или зашумленными данными, трудностями в предсказании редких событий. Методы экстраполяции не позволяют сделать такие предсказания. Установка и обслуживание датчиков стоят недешево, датчики подвержены ошибкам и сбоям, могут давать не-

*Таблица 1*

**Сравнительный анализ моделей цифровых двойников организационно-технических систем**

| Характеристика | Подход к моделированию | | |
|---|---|---|---|
| | Математическое моделирование | Моделирование, основанное на данных | Гибридное моделирование |
| Способ описания системы | Описывает законы функционирования объекта (процесса) и его связи с внешней средой. Моделируется поведение системы, выявляются причинно-следственные связи и закономерности | Строится на основании имеющихся эмпирических данных с применением инструментов машинного обучения. Задача построения модели сводится к подбору параметров модели и композиции функций из некоторого семейства | Строится на основе законов функционирования и настраивается (адаптируется) с учетом эмпирических данных |
| Принцип моделирования | Модель "белого ящика", моделирование причинно-следственных связей | Модель "черного ящика", моделирование корреляций | Модель "серого ящика" |
| Направление моделирования | Сверху вниз | Снизу вверх | Сверху вниз, снизу вверх |
| Описание и степень определенности информации | Неопределенность информации контролируется входными данными и точностью моделирования. Описание — детерминированное, вероятностное | Вероятностное описание информации на основе распределений данных в обучающих выборках | Детерминированное, вероятностное |
| Методы моделирования | Численные методы, методы исследования операций, методы имитационного и ситуационного моделирования | Статистические методы, методы экстраполяции, методы машинного обучения, методы аналитики больших данных | Междисциплинарные модели |
| Прогностическая способность | Прогнозирование в широких интервалах значений параметров, описываемых моделью | Трудность в предсказании редких событий, а также в условиях неполных данных и зашумленной информации, а также за пределами обучающих выборок | Высокая прогностическая способность в пределах штатных/внештатных ситуаций |
| Приоритетный подход к принятию решений и управлению | Принятие решений основано на анализе совокупной производительности (эффективности) системы. Формирование управленческих решений на основе решения обратных задач | Принятие решений основано на анализе данных мониторинга, диагностики. Формирование управленческих решений на основе прогнозирования и решения прямых задач | Решение как прямых, так и обратных задач управления |
| Тип системы управления | Управление по отклонению, управление с адаптацией | Управление по отклонению, управление с адаптацией | Управление по отклонению с учетом слабых сигналов среды; рефлексивное управление |
| Этап жизненного цикла системы | Все стадии | Эксплуатация | Рост, стабильность |
| Схема работы | Численное моделирование + датчики→сбор данных→ IIoT*-платформа | Датчики + IIoT-платформа→ сбор данных→аналитика данных | Математическое моделирование + датчики→сбор данных→ IIoT-платформа→аналитика |
| Инструментальные средства | Matlab Simulink, ANSYS, AnyLogic, Ithink и др. | R, Python, Statictica, GPSS и др. | Междисциплинарные платформы |

* IIoT — Industrial Internet of Things.

правильные показания, а результаты могут перегружать пользователей избыточной информацией.

Без структурной (математической, физической) модели трудно определить области технических устройств, где целесообразно располагать датчики. Сбор исходных данных с датчиков является только частью процесса моделирования. На этапе, когда появляется обратная задача, т. е. когда необходимо на основе данных, полученных с датчиков, восстановить картину происходящего, без математической модели эта задача оказывается трудноразрешимой, так как большинство собранных данных является непригодными, "мусорными", из которых очень сложно выделить содержательную часть, адекватно описывающую объект (процесс).

При этом математическое моделирование объектов (процессов) в сочетании с моделями, основанными на данных, дает больше возможностей для прогнозирования, чем модели, основанные только на базе технологий машинного обучения. Моделирование, основанное на данных, как правило, ограничено лишь этапом эксплуатации изделия. Математические модели, основанные на физических процессах, более перспективны в задачах ситуационного анализа и для принятия решений в условиях, заданных выражением "что будет, если?". Кроме этого, гибридные модели могут использоваться в неповторяющихся ситуациях, когда нет достаточных данных для применения статистических методов.

На базе дополнительной информации, полученной на этапе эксплуатации, повышается уровень адекватности гибридной модели, т. е. ЦД обучается и позволяет в дальнейшем прогнозировать уровень возможных отклонений от штатных режимов, повреждений оборудования или оценить его остаточный ресурс [7].

Следует отметить, что на разных этапах создания цифровой модели объекта существует разный объем данных о поведении физического объекта. На этапе разработки данных от реального объекта нет, поскольку нет самого физического изделия (продукта), и данные об объекте могут быть получены только на основе моделирования физических процессов, определяющих создание и функционирование будущего изделия. По мере накопления данных об изделии, последние все в большей мере могут использоваться для построения аналитических моделей. На этапе тестирования и эксплуатации готового изделия появляется его "цифровая тень". Математическая модель на базе физических процессов может создаваться до этапа создания реального объекта и предсказывать его поведение в широких пределах при изменении краевых условий задачи численного моделирования.

Представленное обобщение существующих подходов проектирования ЦД организационно-технических систем позволяет, с одной стороны, систематизировать методы и модели, используемые для построения ЦД таких систем с учетом требований к системе управления описания неопределенности параметров системы, стадии ее жизненного цикла и других существенных свойств, а, с другой стороны, осуществлять выбор инструментария моделирования, адекватного объекту управления и его особенностям.

Цифровые двойники с высоким уровнем адекватности должны комбинировать как модели физических процессов, так и модели, основанные на данных. Умные ЦД с интеллектуальным управлением должны совмещать оба подхода, усиливая преимущества каждого из них.

### 1.2. Приложения цифровых двойников

Использование и сфера приложений ЦД очень широки. Они используются в разнообразных организационно-технических системах: в тяжелом машиностроении [11—13], в автомобилестроении [14] и судостроении [15], в области атомной энергетики [16], в авиакосмической [17, 18] и нефтегазовой отраслях [19], в архитектурном проектировании и создании "умных" городов [20], в сельском хозяйстве, а также их применяют для повышения операционной эффективности при производстве потребительских товаров [3], точности диагностики и принятии решений в здравоохранении [21, 22], для привлечения клиентов и кастомизации услуг в финансовом секторе [23] и в ретейле [24], для организации логистических процессов и цепочек поставок [25], в региональном и муниципальном менеджменте [26, 27].

Востребованность и спектр применения ЦД расширяются. Так как их разработка и реализация базируются на ряде стремительно развивающихся технологий, то развитие ЦД напрямую зависит от роста возможностей этих технологий. Это обусловлено перечисленными далее факторами.

1. Развитием квантовых технологий и ростом быстродействия вычислительных систем [28, 29]. С переходом к квантовым вычислениям прогнозируется качественный скачок быстродействия аппаратных систем в ближайшее десятилетие. Это обстоятельство позволит выполнять численный анализ на основе уже существующих (и более сложных) моделей за время, приемлемое для оперативного взаимодействия физического объекта и его цифровой копии. Сегодня компании работают над разработкой и использованием квантовых алгоритмов для моделирования сложных физи-

ческих процессов. Переход к таким технологиям позволит ускорить решение задач, основанных на численном моделировании, обеспечивая требуемую точность алгоритмов в условиях доступных вычислительных ресурсов (задачи многопараметрической оптимизации и др.).

2. Развитием технологии 5G [30—32]. Эта технология обладает более высокой пропускной способностью, меньшим временем задержки, меньшим расходом энергии батарей IoT-датчиков. Это обеспечивает рост скорости передачи сигналов между физическим объектом и его ЦД. Применение сетей 5G позволит сконструировать сервисы виртуальной реальности в составе ЦД и сделать доступной виртуальную верификацию и валидацию готовых продуктов.

3. Развитие технологии сильного искусственного интеллекта [33, 34] позволят строить ЦД, в которых роль человека в принятии управленческих решений будет сведена к минимуму. Цифровые двойники смогут обеспечить принятие решений автономно, координировать эти решения с другими ЦД, выполнять самотестирование и диагностику с последующим устранением неисправностей. Такие системы поддержки принятия решения на базе ЦД обеспечат принятие сложных решений в агрессивных и опасных средах без присутствия человека.

## 2. Методология системного инжиниринга и проектирования цифровых двойников организационно-технических систем

Процесс построения ЦД является многостадийным и состоит из следующих стадий: концептуализация, проектирование, цифровое моделирование и технологические испытания (рис. 1).

Цифровой двойник определяется как система, состоящая из цифровой модели физического объекта и двусторонних информационных связей с физическим объектом или его компонентами. В основе ЦД лежит цифровая модель в виде математических и компьютерных моделей, а также документов, описывающих структуру, функциональные возможности и поведение вновь разрабатываемого или эксплуатируемого объекта на разных этапах его жизненного цикла. По результатам цифровых или иных испытаний проводится оценка соответствия готовой продукции определенным требованиям. Цифровая модель описывает структуру, функции и поведение разрабатываемого физического объекта. Содержание и функциональность цифровой модели зависят от стадии жизненного цикла физического объекта. Оценка соответствия цифровой модели физического объекта включает процедуры проверки и валидации математических и компьютерных моделей.

Организационно-методическое обеспечение процесса разработки и использования ЦД является не до конца проработанным с точки зрения согласования задач по описанию объекта моделирования и управления (физического объекта), в том числе структурного, функционального, информационного, а также формирования работ в рамках методологии управления проектами (в том числе гибкой методологии Agile, Scrum-подхода и др.). Для восполнения этого пробела предлагается сформировать план работ по этапам в виде триады: задача этапа — содержание этапа — результаты этапа. Для организации такой работы требуются применение системного подхода и проектирование всех этапов жизненного цикла объекта, включая идентификацию и решение возникающих проблем в процессе его функционирования.

Автором настоящей статьи разработана методология, обеспечивающая организационно-методическую поддержку процесса разработки и эксплуатации ЦД организационно-технической системы, представленная в табл. 2. Организация процесса проектирования объединяет этапы его проектирования, методы и модели, а также обеспечивает ускоренный инжиниринг ЦД.

Принципиально важно, что предлагаемая методология учитывает специфику объекта управления — организационно-технической системы как

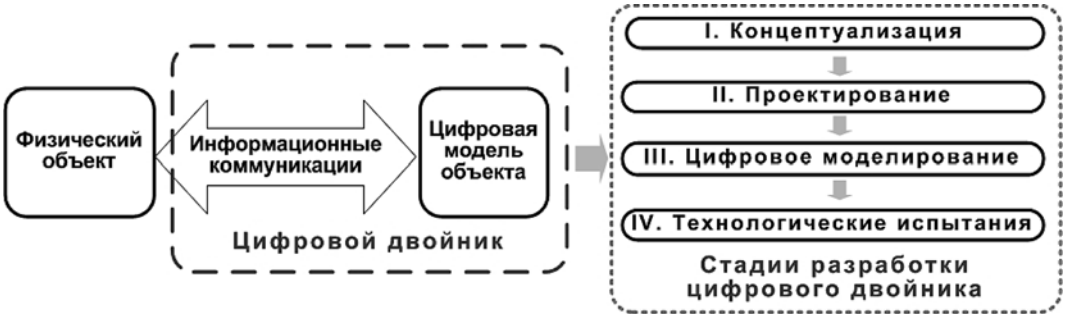


**Рис. 1. Цифровой двойник и стадии его разработки**

*Таблица 2*

**Методология разработки цифрового двойника объекта (процесса)**

| Стадия | Этап и задача этапа | Содержание этапа | Результаты этапа |
|---|---|---|---|
| 1. Концептуализация | 1.1. Выявление противоречий | 1.1.1. Мониторинг и анализ противоречий между текущим и желаемым состояниями объекта | Противоречия, проблемы |
| | 1.2. Определение целей и выбор критериев | 1.1.2. Формулировка целей развития объекта | Цели, критерии эффективности результатов функционирования объекта |
| 2. Проектирование | 2.1. Декомпозиция (сканирование) объекта | 2.1.1. Сбор исходной информации об объекте моделирования | Статистическая информация. Обзор литературы |
| | | 2.1.2. Статистический анализ состояния и динамики развития объекта моделирования на основе причинного и факторного анализа | Основные тренды и факторы влияния |
| | | 2.1.3. Функциональная, структурная, информационная декомпозиция объекта, декомпозиция по жизненному циклу (на основе SADT-, ARIS-, BPMN-методологии) | Функциональная модель<br>Организационная модель<br>Информационная модель<br>Модель жизненного цикла |
| | 2.2. Анализ окружения объекта | 2.2.1. Анализ факторов внешней и внутренней среды, оказывающих влияние на функционирование объекта, на основе STEP- и SWOT-анализа | Ранжирование факторов по степени важности |
| | | 2.2.2. Определение возможных перспектив развития объекта на основе Delphi-анализа | Прогнозные оценки результатов, сроков, положения научных результатов и мероприятий поддержки внедрения новых разработок. Выбор важнейших направлений исследований |
| | | 2.2.3. Построение Wild-карт | Определение возможных событий, которые могут кардинально изменить вероятный ход событий |
| | | 2.2.4. Выявление стейкхолдеров и построение дорожной карты | Матрицы стейкхолдеров, дорожная карта для среднесрочного развития объекта |
| | 2.3. Синтез альтернативных решений проблем (на качественном уровне) | 2.3.1. Описание результатов проекта в терминах существующих сильных и слабых сторон и будущих возможностей и угроз на базе SWOT-анализа | Альтернативы решения проблем |
| | | 2.3.2. Определение альтернативных путей достижения целей на основе сценарного анализа | Описание представления системы (объекта) в будущем |
| 3. Цифровое моделирование | 3.1. Выбор средств математического и компьютерного моделирования и защиты данных | 3.1.1. Обоснование выбора математических методов и моделей формализации объекта | Математические методы формализации объекта |
| | | 3.1.2. Обоснование выбора программного обеспечения | Программное обеспечение |
| | 3.2. Построение математической модели (моделирование, оценка и оптимизация) | 3.2.1. Построение математической модели объекта, валидация и оценка адекватности | Математическая модель объекта |
| | | 3.2.2. Решение задачи синтеза оптимальных решений. Исследование устойчивости и адекватности решений | Оптимальные решения |
| | 3.3. Построение компьютерной модели | 3.3.1. Написание исходного кода программы | Первичный программный код |
| | | 3.3.2. Отладка, тестирование, верификация кода на исходных данных | Компьютерная модель объекта |
| | 3.4. Построение системы поддержки принятия решений | 3.4.1. Бесшовная интеграция блока управления в цифровую модель ЦД | Система поддержки принятия решений |

*Продолжение табл. 2*

| Стадия | Этап и задача этапа | Содержание этапа | Результаты этапа |
|---|---|---|---|
| 4. Технологические испытания | 4.1. Построение виртуального полигона и проведение стендовых экспериментов | 4.1.1. Формирование испытательного стенда | Виртуальный полигон: система для проведения стендовых испытаний (технические средства, программное, методическое и организационное обеспечение) |
| | | 4.1.2. Проведение виртуальных экспериментов | Количественные и качественные характеристики объекта в результате экспериментов |

системы междисциплинарной природы, объединяющей техническую (производственную) систему и организационную систему (человека). Организационно-техническая система имеет следующие особенности:

- самостоятельное целеполагание, целенаправленность поведения, в результате чего может возникнуть сознательное искажение информации, невыполнение требуемых обязательств;
- рефлексия и прогнозирование поведения объекта/субъекта управления;
- ограниченная рациональность, в результате чего обеспечивается принятие решений в условиях неопределенности и ограничений на объем обрабатываемой информации.

Предложенный методологический подход позволит, во-первых, осуществлять системный анализ объекта моделирования и управления с учетом неопределенности внешней среды на базе разнородных инструментальных средств качественного и количественного анализа. Во-вторых, он предоставит возможность сформировать адекватную математическую модель объекта с учетом результатов этапа концептуализации и выработать компьютерную модель и осуществить ее испытания. В-третьих, этот подход может явиться основанием для построения ЦД объекта/процесса и формирования системы поддержки принятия решений.

При рассмотрении физического объекта на стадии "разработка" его жизненного цикла необходим прототип ЦД, содержащий необходимые компоненты для описания и создания физической версии объекта. На этом этапе задача состоит также в том, чтобы предвидеть возможные состояния объекта и разработать систему поддержки принятия решений для нейтрализации последствий непредвиденных и нежелательных событий.

В отличие от традиционных подходов, сводящихся к проверке и подтверждению требований к разрабатываемому объекту, а также устранению проблем и сбоев для прогнозируемых состояний объекта, прототип ЦД может помочь выявлению и устранению непредвиденных нежелательных состояний. Эта задача решается на основе изменения параметров моделирования в возможных пределах и исследования множества различных ситуаций и разнообразных поведенческих паттернов, которые могут привести к серьезным катастрофическим проблемам. Такое моделирование позволит спроектировать физический объект в виртуальном пространстве с набором новых возможностей и значительно уменьшить риски нежелательного и непредсказуемого поведения объекта, а также устранить негативные последствия таких рисков.

В данной работе рассмотрены только следующие стадии построения ЦД: конструирование и проектирование, стадии 1—2, а также этап 3.4 "Разработка системы поддержки принятия решений". В этом контексте схема функционирования предложенного методологического подхода включает следующие шаги. Сначала необходимо провести идентификацию проблем и описать противоречия, которые возникают при разработке и внедрении ЦД в промышленность. Далее следует определить цели внедрения ЦД, поставить задачи исходя из этой цели и описать проект. На втором шаге происходит декомпозиция проектируемого физического объекта. Описываются функции и свойства, а также технические параметры. Далее строятся его структурная и функциональная модели. Третий шаг посвящен анализу внешней среды функционирования объекта. С помощью STEP- и SWOT-анализа определяются угрозы и возможности внешней среды и экспертная оценка их влияния на эффективность ЦД. На четвертом шаге осуществляется построение системы поддержки принятия решений при возникновении неисправностей системы на основе выбранной математической модели и проводятся имитационные эксперименты.

## 3. Численный пример разработки прототипа цифрового двойника

В качестве примера рассмотрим техническое устройство — инкубатор интенсивной терапии новорожденных с микропроцессорным управлением мониторинга параметров температуры, кон-

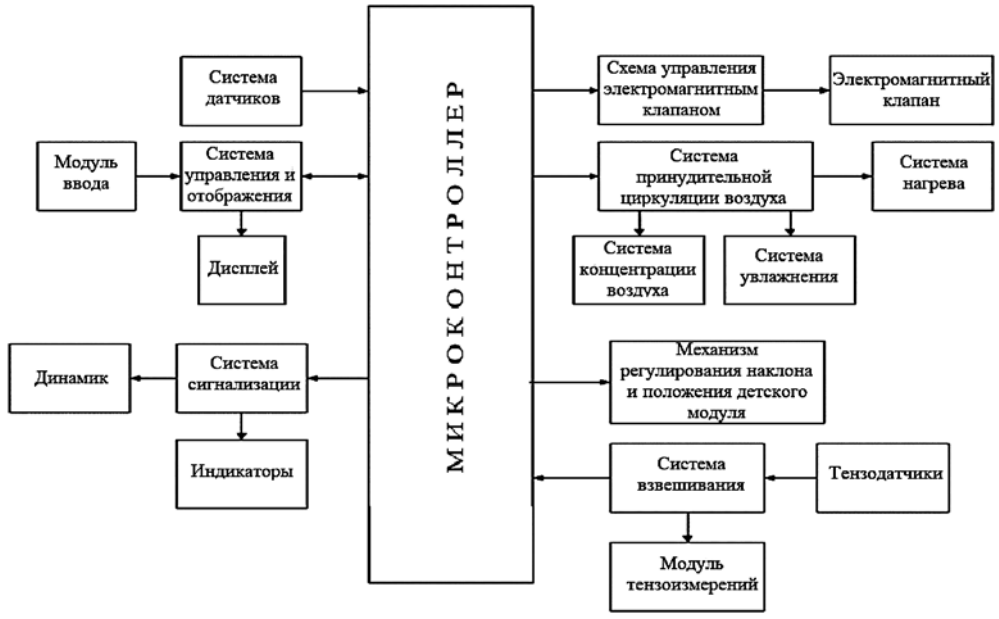


**Рис. 2. Структурная схема технического устройства**

центрации кислорода, влажности воздуха, температуры и массы тела новорожденного (аналог ИДН-03-"УОМЗ" [35]). Инкубатор предназначен для выхаживания и проведения интенсивной терапии новорожденных, в том числе недоношенных с критически малой массой (от 500 г). Инкубатор обеспечивает регулируемый приток теплоты, требуемую влажность воздуха и концентрацию кислорода в детском модуле, контроль массы тела.

Сначала сформируем структурную и функциональную модели устройства. Структурная схема представлена на рис. 2 и состоит из системы датчиков, системы управления оборудованием, системы регулирования температуры и подачи кислорода[1]. Наблюдаемыми параметрами устройства являются: а) температура воздуха; б) температура кожи; в) относительная влажность воздуха; г) концентрация кислорода; д) масса тела.

Построение функциональной модели позволяет четко зафиксировать, какие процессы осуществляются, какие информационные объекты используются при выполнении функций различного уровня детализации. Модель показывает зоны ответственности исполнителей процесса и ход самого процесса, связи процессов между собой и результаты выполнения процессов. Функциональная модель является основой для выявления проблем и "узких" мест функционирования прибора.

Функциональная модель процесса функционирования инкубатора формируется на основе нотации системного моделирования бизнес-процессов IDEF0. На рис. 3 продемонстрирован первый уровень декомпозиции и отражены выполняемые функции прибора.

На следующем шаге формируется диаграмма причинно-следственных связей факторов для выявления возможных причин неисправностей оборудования. Выявлено, что основными источниками возникновения неисправности инкубатора являются следующие шесть факторов: 1) медицинский персонал; 2) технический персонал; 3) внешняя среда; 4) система датчиков; 5) система управления; 6) система питания. На диаграмме представлена декомпозиция этих факторов в виде проблем, которые, действуя изолированно или совместно, могут повлечь неисправность оборудования (рис. 4).

## 4. Модель поддержки принятия решений для диагностики технического состояния оборудования на основе методов нечеткой логики

Выявленные проблемы и возможные причины неисправности оборудования показывают, что

[1] В систему датчиков устройства входят: датчики температуры воздуха (основной и дополнительный); датчики температуры кожи (основной и дополнительный); датчики влажности (регулирующий и отображающий); датчики концентрации кислорода (регулирующий и отображающий). Регулирование температуры, влажности и концентрации кислорода осуществляется с помощью системы принудительной циркуляции воздуха. В инкубаторе применяют два режима автоматического регулирования температуры: по воздуху и по коже ребенка. Система сигнализации состоит из динамиков, индикатора центральной сигнализации, красного индикатора тревоги и красного индикатора нарушения подачи электроэнергии. Электромагнитный клапан предназначен для насыщения воздуха кислородом. Поступающий в систему подачи воздуха кислород нагревается и увлажняется вместе с воздухом.

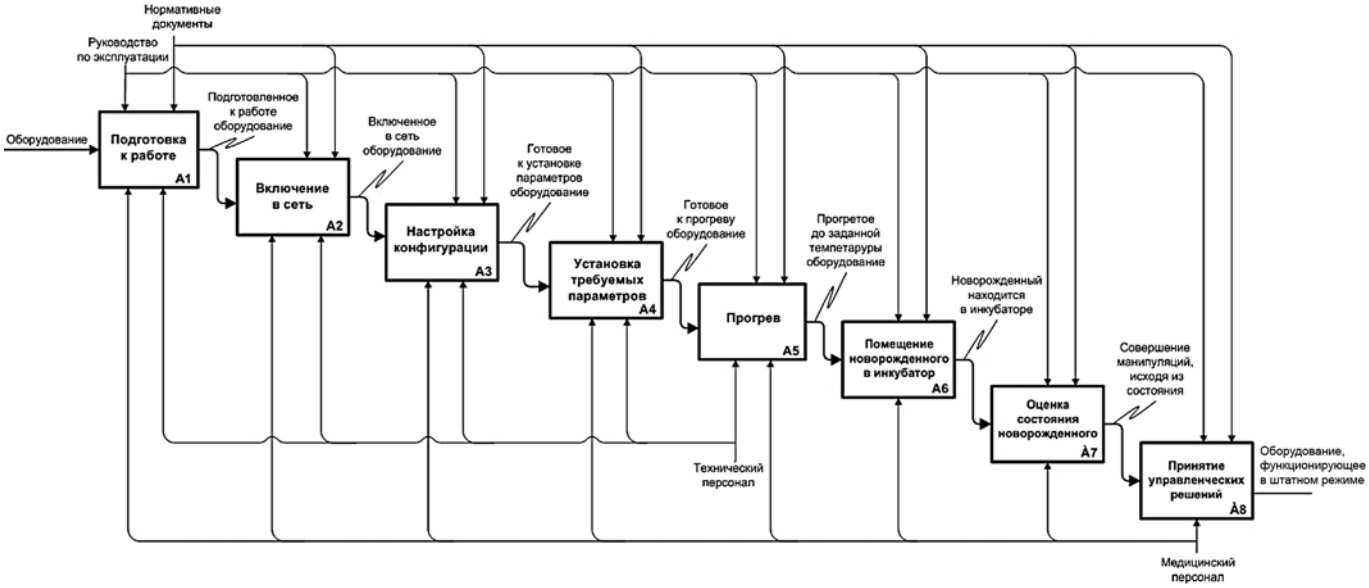


**Рис. 3. Функциональная схема работы технического устройства (первый уровень декомпозиции)**

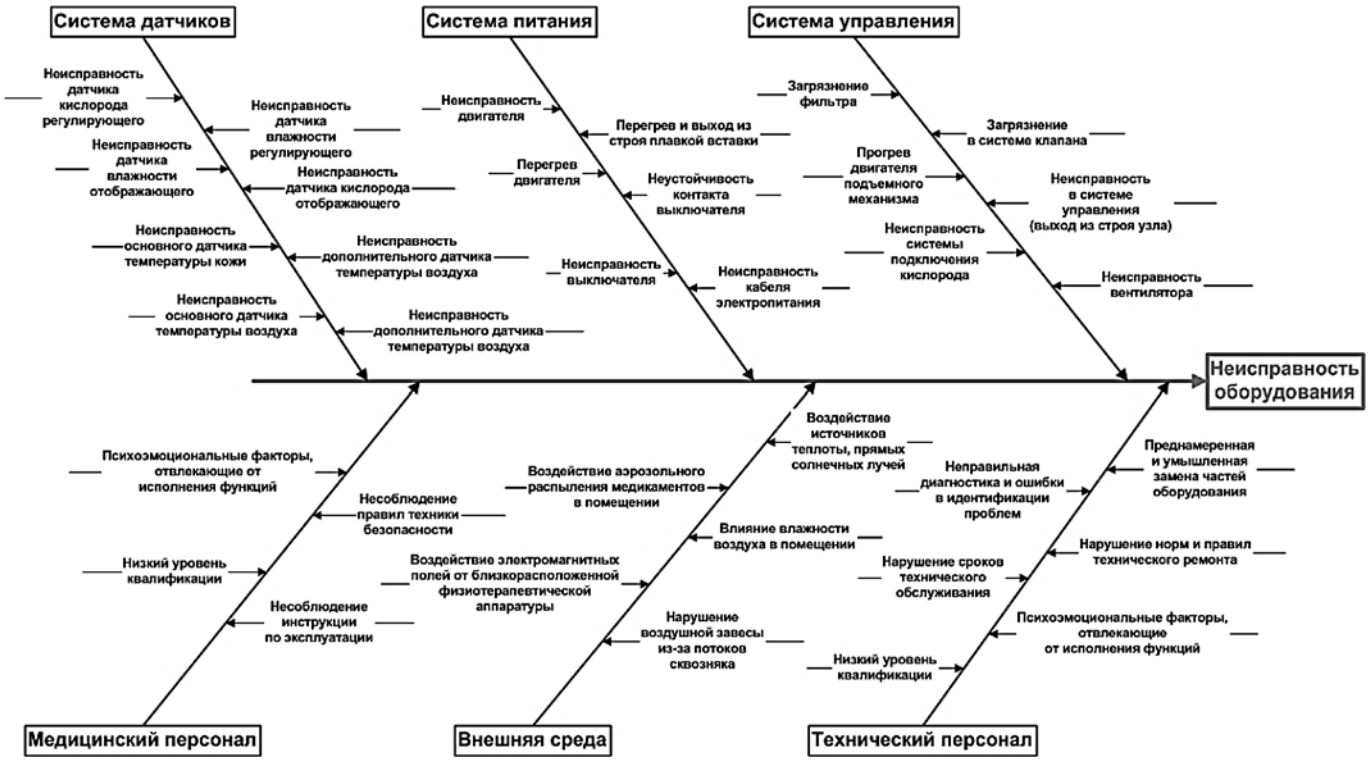


**Рис. 4. Диаграмма причинно-следственных связей факторов появления неисправностей**

ряд факторов и причин описываются не в количественной, а в качественной форме. Поэтому требуется система поддержки принятия решений, которая учитывает разные шкалы измерения моделируемых свойств оборудования. Задачи такого типа можно решать с применением методов искусственного интеллекта и моделей нечеткой логики.

Категория технического состояния зависит от большого числа факторов, как качественных, так и количественных. Разработанная модель рассматривает факторы неисправности, которые связаны с внутренним техническим состоянием самого оборудования. Исследуются неисправности, связанные с двигателем детского модуля, кабелем

электропитания, системой подключения кислорода, системой клапана, крыльчаткой вентилятора, датчиком влажности, датчиками температуры кожи основным и дополнительным и т. д.

Реализация процесса нечеткого моделирования проводится посредством применения модуля Fuzzy Logic Toolbox программного средства MATLAB. Выполнение нечеткого вывода реализуется на основе алгоритма Мамдани *(Mamdani)*.

Для решения задачи диагностики технического состояния оборудования требуется качественное ее описание на основе лингвистических выражений (логических правил). Используются 25 входных переменных, отражающих состояние подсистем оборудования, детерминирующих одно из четырех возможных решений по действию с оборудованием — вывести из эксплуатации, провести ремонт, провести дополнительную подготовку к работе, оставить в эксплуатации (табл. 3).

Лингвистические переменные и область их возможных значений показаны в табл. 4.

Функции принадлежности для лингвистических переменных представлены на рис. 5, см. четвертую сторону обложки.

База правил для принятия решений по техническому состоянию и действию с оборудованием при возникновении неисправностей определяется

*Таблица 3*

**Описание входных и выходных переменных**

| Обозначение переменной | Описание | Принимаемые значения |
|---|---|---|
| **Тип переменной — входная** | | |
| $I_1$ | Регулирование положения детского модуля по высоте | Невозможно, возможно |
| $I_2$ | Приведение детского модуля в наклонное положение | Невозможно, возможно |
| $I_3$ | Двигатель детского модуля | Неисправен, перегрев, исправен |
| $I_4$ | Выключатель механизма детского модуля | Неисправен, исправен |
| $I_5$ | Механизм регулирования положения детского модуля по высоте | Отключается, не отключается |
| $I_6$ | Температура воздуха под колпаком | Пониженная, нормальная, повышенная |
| $I_7$ | Фильтр | Грязный, чистый |
| $I_8$ | Срок установки фильтра | Более 3 месяцев, менее 3 месяцев |
| $I_9$ | Вода в бачке увлажняющей системы | Отсутствует, присутствует |
| $I_{10}$ | Система клапана | Загрязнена, чистая |
| $I_{11}$ | Индикатор | Красный, сеть, центральной сигнализации |
| $I_{12}$ | Кабель электропитания | Не присоединен, присоединен |
| $I_{13}$ | Звуковой сигнал | "3", "2", прерывистый, непрерывный |
| $I_{14}$ | Крыльчатка вентилятора | Установлена неправильно, установлена правильно |
| $I_{15}$ | Вентилятор | Неисправен, исправен |
| $I_{16}$ | Датчик влажности отображающий | Неисправен, исправен |
| $I_{17}$ | Датчик влажности регулирующий | Неисправен, исправен |
| $I_{18}$ | Датчик температуры кожи основной | Неисправен, исправен, не присоединен, присоединен |
| $I_{19}$ | Датчик температуры кожи дополнительный | Неисправен, исправен, не присоединен, присоединен |
| $I_{20}$ | Концентрация кислорода воздуха | Низкая, средняя, высокая |
| $I_{21}$ | Система подключения кислорода | Неисправна, исправна |
| $I_{22}$ | Система управления | Неисправна, исправна |
| $I_{23}$ | Моральный износ | Отсутствует, присутствует |
| $I_{24}$ | Физический износ | Неустранимый, устранимый |
| $I_{25}$ | Плавкая вставка | Перегорела, не перегорела |
| **Тип переменной — выходная** | | |
| $O_1$ | Техническое состояние и действие с оборудованием | Вывести из эксплуатации, провести ремонт, провести дополнительную подготовку к работе, оставить в эксплуатации |

*Таблица 4*

**Описание лингвистических переменных**

| Переменная | Значение переменной | Диапазон значений |
|---|---|---|
| $I_1$, $I_2$, $I_4$, $I_5$, $I_7$, $I_8$, $I_9$, $I_{10}$, $I_{12}$, $I_{14}$, $I_{15}$, $I_{16}$, $I_{17}$, $I_{21}$, $I_{22}$, $I_{23}$, $I_{25}$ | "Невозможно", "Неисправен", "Не отключается", "Грязный", "Более 3 месяцев", "Отсутствует", "Не присоединен", "Установлена неправильно", "Не устраняется" | (0; 0,35; 0,7) |
| $I_1$, $I_2$, $I_4$, $I_5$, $I_7$, $I_8$, $I_9$, $I_{10}$, $I_{12}$, $I_{14}$, $I_{15}$, $I_{16}$, $I_{17}$, $I_{21}$, $I_{22}$, $I_{23}$, $I_{25}$ | "Возможно", "Исправен", "Отключается", "Чистый", "Менее 3 месяцев", "Присутствует", "Присоединен", "Установлена правильно", "Устраняется" | (0,4; 0,7; 1) |
| $I_3$, $I_6$, $I_{11}$, $I_{20}$ | "Пониженная", "Красный", "Низкая", "Неисправность" | (0; 0,2; 0,4) |
| $I_3$, $I_6$, $I_{11}$, $I_{20}$ | "Нормальная", "Сеть", "Средняя", "Перегрев" | (0,3; 0,5; 0,7) |
| $I_3$, $I_6$, $I_{11}$, $I_{20}$ | "Повышенная", "Высокая", "Центральной сигнализации", "Исправность" | (0,6; 0,8; 1) |
| $I_{13}$, $I_{18}$, $I_{19}$, $O_1$ | "Вывести из эксплуатации", "Неисправен" | (0; 0,175; 0,35) |
| $I_{13}$, $I_{18}$, $I_{19}$, $O_1$ | "Провести ремонт", "Исправен" | (0,2; 0,375; 0,55) |
| $I_{13}$, $I_{18}$, $I_{19}$, $O_1$ | "Прерывистый", "Провести дополнительную подготовку к работе", "Не присоединен" | (0,4; 0,575; 0,75) |
| $I_{13}$, $I_{18}$, $I_{19}$, $O_1$ | "Непрерывный", "Оставить в эксплуатации", "Присоединен". | (0,65; 0,825; 1) |

с помощью набора продукционных правил вида If-Then (выборочно):

— If ($I_1$ is "возможно") and ($I_2$ is "возможно") and ($I_3$ is "исправность") and ($I_4$ is "неисправен") and ($I_5$ is "не отключается") then ($O_1$ is "провести ремонт");

— If ($I_1$ is "возможно") and ($I_2$ is "возможно") and ($I_3$ is "исправность") and ($I_4$ is "исправен") and ($I_5$ is "отключается") then ($O_1$ is "оставить в эксплуатации");

— If ($I_7$ is "чистый") and ($I_9$ is "присутствует") then ($O_1$ is "оставить в эксплуатации");

— If ($I_{11}$ is "красный") and ($I_{13}$ is "2") and ($I_{20}$ is "низкая") and ($I_{21}$ is "исправна") then ($O_1$ is "провести ремонт");

— If ($I_{11}$ is "красный") and ($I_{13}$ is "2") and ($I_{20}$ is "низкая") and ($I_{21}$ is "исправна") then ($O_1$ is "провести дополнительную подготовку к работе");

— If ($I_{20}$ is "средняя") and ($I_{21}$ is "исправна") then ($O_1$ is "оставить в эксплуатации");

— If ($I_{11}$ is "центральной сигнализации") and ($I_{13}$ is "3") and ($I_{20}$ is "средняя") and ($I_{22}$ is "неисправна") then ($O_1$ is "провести ремонт");

— If ($I_{23}$ is "присутствует") and ($I_{24}$ is "неустранимый") then ($O_1$ is "вывести из эксплуатации");

— If ($I_{23}$ is "отсутствует") and ($I_{24}$ is "неустранимый") then ($O_1$ is "вывести из эксплуатации");

— If ($I_1$ is "невозможно") and ($I_2$ is "возможно") and ($I_3$ is "перегрев") and ($I_4$ is "исправен") and ($I_5$ is "отключается") then ($O_1$ is "провести дополнительную подготовку к работе").

Аккумулирование заключения по всем правилам проведено с применением операции max-дизъюнкции. При дефаззификации использован метод центра тяжести. Реализуя систему нечеткого вывода на этапе дефаззификации, получим решение по режиму эксплуатации оборудовании при возникновении неисправностей в условиях известных входных данных.

Для анализа, учета и принятия решений по идентификации технического состоянию и действию с оборудованием в условиях различных ситуаций, определяемых сочетанием входных переменных, проведен ситуационный анализ. Рассматриваются многовариантные ситуации с разным набором сочетаний входных факторов следующих типов: 1) горит световой индикатор "сеть", звучит непрерывный сигнал; 2) мигает красный индикатор тревоги, звучит звуковой сигнал "3"; 3) не функционирует детский модуль. Выборочные результаты моделирования решений по этим ситуациям представлены в табл. 5.

В ситуации, при которой горит световой индикатор "сеть"; звучит непрерывный сигнал и плавкая вставка перегорела, требуется срочный ремонт оборудования. Если мигает красный индикатор тревоги, звучит звуковой сигнал "3" и датчик температуры кожи основной присоединен, а датчик температуры кожи дополнительный не присоединен, необходимо провести дополнительную подготовку оборудования к работе. Если механизм регулирования положения детского модуля по высоте не отключается и выключатель неисправен и двигатель модуля неисправен, то необходимо вывести оборудование из эксплуатации.

Разработанная модель принятия решений по выявлению и устранению неисправностей оборудования в составе его ЦД учитывает изменения внутренних и внешних факторов функционирования оборудования и позволяет в кратчайшие

*Таблица 5*

**Результаты ситуационного анализа**

| Ситуации типа 1 | | Входные переменные | | | | | Выходная переменная |
|---|---|---|---|---|---|---|---|
| | | $I_{11}$ | $I_{13}$ | $I_{12}$ | $I_{25}$ | Все остальные | |
| 1. Горит световой индикатор "сеть"; звучит непрерывный сигнал | 1.1. Кабель электропитания не присоединен | 0,5 | 0,8 | 0,2 | 0,7 | 0,7 | **0,59** |
| | 1.2. Кабель электропитания присоединен | 0,5 | 0,8 | 0,8 | 0,8 | 0,7 | **0,92** |
| | 1.3. Плавкая вставка перегорела | 0,5 | 0,8 | 0,1 | 0,35 | 0,7 | **0,376** |
| | 1.4. Плавкая вставка не перегорела | 0,5 | 0,8 | 0,9 | 0,2 | 0,7 | **0,95** |
| Ситуации типа 2 | | Входные переменные | | | | | Выходная переменная |
| | | $I_{11}$ | $I_{13}$ | $I_{18}$ | $I_{19}$ | Все остальные | |
| 2. Мигает красный индикатор тревоги, звучит звуковой сигнал "3" | 2.1. Датчик температуры кожи основной не присоединен, датчик температуры кожи дополнительный присоединен | 0,1 | 0,3 | 0,5 | 0,8 | 0,6 | **0,74** |
| | 2.2. Датчик температуры кожи основной присоединен, датчик температуры кожи дополнительный не присоединен | 0,1 | 0,3 | 0,9 | 0,6 | 0,6 | **0,65** |
| | 2.3. Датчик температуры кожи основной не присоединен, датчик температуры кожи дополнительный не присоединен | 0,1 | 0,3 | 0,7 | 0,55 | 0,6 | **0,574** |
| | 2.4. Датчик температуры кожи основной неисправен, датчик температуры кожи дополнительный неисправен | 0,1 | 0,3 | 0,1 | 0,2 | 0,6 | **0,454** |
| | 2.5. Датчик температуры кожи основной исправен, датчик температуры кожи дополнительный неисправен. | 0,1 | 0,3 | 0,35 | 0,1 | 0,6 | **0,89** |
| | 2.6. Датчик температуры кожи основной неисправен, датчик температуры кожи дополнительный исправен | 0,1 | 0,3 | 0,3 | 0,4 | 0,6 | **0,95** |

| Ситуации типа 3 | | Входные переменные | | | | | | Выходная переменная |
|---|---|---|---|---|---|---|---|---|
| | | $I_1$ | $I_2$ | $I_3$ | $I_4$ | $I_5$ | Все остальные | |
| 3. Не функционирует детский модуль | 3.1. Механизм регулирования положения детского модуля по высоте не отключается, выключатель не исправен, двигатель исправен | 1 | 1 | 0,7 | 0,2 | 0,4 | 0,5 | **0,464** |
| | 3.2. Механизм регулирования положения детского модуля по высоте не отключается, выключатель исправен, двигатель неисправен | 1 | 1 | 0,1 | 0,7 | 0,3 | 0,5 | **0,52** |
| | 3.3. Механизм регулирования положения детского модуля по высоте не отключается, выключатель неисправен, двигатель неисправен | 1 | 1 | 0,3 | 0,5 | 0,2 | 0,5 | **0,34** |
| | 3.4. Регулирование положения детского модуля по высоте невозможно, перегрев двигателя | 0,25 | 1 | 0,5 | 0,9 | 1 | 0,5 | **0,62** |
| | 3.5. Регулирование положения детского модуля по высоте невозможно, двигатель не исправен | 0,3 | 1 | 0,1 | 0,9 | 1 | 0,5 | **0,376** |
| | 3.6. Регулирование положения детского модуля по высоте невозможно, двигатель исправен | 0,1 | 1 | 0,8 | 0,9 | 1 | 0,5 | **0,75** |
| | 3.7. Приведение детского модуля в наклонное положение невозможно, двигатель неисправен | 1 | 0,5 | 0,2 | 1 | 0,8 | 0,5 | **0,45** |
| | 3.8. Приведение детского модуля в наклонное положение невозможно, перегрев двигателя | 1 | 0,2 | 0,6 | 1 | 0,8 | 0,5 | **0,68** |
| | 3.9. Приведение детского модуля в наклонное положение невозможно, двигатель исправен | 1 | 0,6 | 0,9 | 1 | 0,8 | 0,5 | **0,7** |

сроки идентифицировать причину неисправности и выработать решение по быстрому переводу оборудования в штатный режим работы. Такие меры позволят сократить простои, уменьшить затраты на ремонтные работы и повысить операционную эффективность оборудования.

## Заключение

В работе проведена систематизация подходов, методов проектирования и моделей ЦД организационно-технических систем. Предложенное обобщение этих подходов может быть полезным для выбора инструментария моделирования, адекватного объекту управления, с учетом его особенностей. Показано, что для сложных систем междисциплинарной природы, таких как организационно-технические системы, обладающих свойствами самостоятельного целеполагания, рефлексии и ограниченной рациональности при принятии решений, не существует комплексного методологического подхода для организации процесса разработки ЦД в целях ускоренного их инжиниринга.

Предложенная методология системного инжиниринга для организации процесса проектирования ЦД организационно-технических систем объединяет этапы проектирования, методы и модели, и обеспечивает ускоренный инжиниринг ЦД объектов/процессов, изготовления готовых изделий (продуктов) высокого качества.

В рамках прототипа ЦД разработана модель поддержки принятия решений для диагностики технического состояния технического устройства, основанная на методах ситуационного анализа и нечеткой логики, обеспечивающая возможность оперировать при принятии решений не только количественными, но и качественными оценками изменений факторов, что обусловливает повышение точности и надежности принимаемых решений. Модель обеспечивает синтез эффективных решений в различных ситуациях и сочетаниях разнородных факторов, характеризующих как состояние оборудования, так и его внешней среды.

Практическую значимость может представлять система поддержки принятия решений на основе разработанной модели, которая является основой для управления при решении проблем, связанных с мониторингом текущего состояния объектов (приборов, оборудования, технологических процессов) и выработкой адекватных решений по устранению их неисправностей. Будучи встроенной в модель ЦД, такая система поддержки принятия решений обеспечит выявление и устранение непредвиденных нежелательных состояний объекта.

### Список литературы


1. **Прохоров А., Лысачев М., Боровков А.** Цифровой двойник. Анализ, тренды, мировой опыт. М.: АльянсПринт, 2020. 401 с.
2. **Петров А. В.** Имитационное моделирование как основа технологии цифровых двойников // Вестник Иркутского государственного технического университета. 2018. Т. 22, № 10. С. 56—66. DOI: 10.21285/1814-3520-2018-10-56-66.
3. **Qi Q., Tao F., Hu T., Anwer N.** et al. Enabling technologies and tools for digital twin // Journal of Manufacturing Systems. 2021. Vol. 58. P. 3—21. DOI: 10.1016/j.jmsy.2019.10.001.
4. **Orlova E. V.** Methodology and Models for Individuals' Creditworthiness Management Using Digital Footprint Data and Machine Learning Methods // Mathematics. 2021. Vol. 9, No. 15. 28 p. DOI: 10.3390/math9151820.
5. **Orlova E. V.** Innovation in Company Labor Productivity Management: Data Science Methods Application // Applied System Innovation. 2021. Vol. 4, No. 3: 68. 18 p. DOI: 10.3390/asi4030068.
6. **Banerjee P.** AI and ML: The Brave New World of Simulation, 2021. URL: https://www.ansys.com/blog/ai-and-ml-the-brave-new-world-of-simulation
7. **Radanliev P., De Roure D., Nicolescu R.** et al. Digital twins: artificial intelligence and the IoT cyber-physical systems in Industry 4.0 // International Journal of Intelligent Robotics and Applications. 2022. Vol. 6, No. 1. P. 171—185. DOI: 10.1007/s41315-021-00180-5.
8. **Орлова Е. В.** Модель оперативного оптимального управления распределением финансовых ресурсов предприятия // Компьютерные исследования и моделирование. 2019. Том 11, № 2. С. 343—358. DOI: 10.20537/2076-7633-2019-11-2-343—358.
9. **Орлова Е. В.** Инженерия системного синтеза эффективности инновационных проектов // Программная инженерия. 2019. Том 10, № 11—12. С. 430—439. DOI: 10.17587/prin.10.430-439.
10. **Орлова Е. В.** Методы и модели анализа данных и машинного обучения в задаче управления производительностью труда // Программная инженерия. 2020. Том 11, № 4. С. 219—229. DOI: 10.17587/prin.11.219-229.
11. **Malik A. A., Masood T., Bilberg A.** Virtual reality in manufacturing: Immersive and collaborative artificial-reality in design of human-robot workspace // International Journal of Computer Integrated Manufacturing. 2019. Vol. 33, No. 1. P. 22—37. DOI: 10.1080/0951192X.2019.1690685.
12. **Tao F., Qi, Q., Wang L., Nee A.** Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison // Engineering. 2019. Vol. 5, No. 4. P. 653—661. DOI: 10.1016/j.eng.2019.01.014.
13. **Mandolla C., Petruzzelli A., Percoco G., Urbinati A.** Building a Digital Twin for Additive Manufacturing through the Exploitation of Blockchain: A case analysis of the aircraft industry // Computers in Industry. 2019. Vol. 109. P. 134—152. DOI: 10.1016/j.compind.2019.04.011.
14. **Korostelkin A. A., Klyavin O. I., Aleshin M. V.** Optimization of Frame Mass in Crash Testing of Off — Road Vehicles // Russian Engineering Research. 2019. Vol. 39, No. 12. P. 1021—1028. URL: https://link.springer.com/article/10.3103/S1068798X19120116
15. **Fonseca Í., Gaspar H., Mello P., Sasaki H.** A Standards-Based Digital Twin of an Experiment with a Scale Model Ship // Computer-Aided Design. 2022. Vol. 145. Article 103191. DOI: 10.1016/j.cad.2021.103191.
16. **Siebert T., Hack E., Labeas G., Patterson E., Splitthof K.** Uncertainty Quantification for DIC Displacement Measurements in Industrial Environments // Experimental Techniques. 2021. Vol. 45, No. 5. P. 685—694. DOI: 10.1007/s40799-021-00447-3.
17. **Future** role of digital twins in the aerospace industry, January 2019. URL: https://www.challenge.org/insights/digital-twin-in-aerospace/
18. **Patterson E., Diamantakos I., Dvurecenska K.** et al. Validation of a structural model of an aircraft cockpit panel: An industrial case study // The Journal of Strain Analysis for Engineering Design. 2021. 030932472110590. DOI: 10.1177/03093247211059084.

19. **Zborowski M.** Finding Meaning, Application for the Much-Discussed "Digital Twin" // Journal of Petroleum Technology. 2018. Vol. 70. P. 26—32. DOI: 10.2118/0618-0026-JPT.

20. **Digital** Twins vs. Building Information Modeling (BIM), 2019. URL: https://www.iotforall.com/digital-twin-vs-bim/

21. **Bruynseels K., Santoni de Sio F., Van Den Hoven J.** Digital Twins in Health Care: Ethical Implications of an Emerging Engineering Paradigm // Frontiers in Genetics. 2018. Vol. 9. DOI: 10.3389/fgene.2018.00031.

22. **Насыров Р. В.** Причинный подход к построению бионических вычислений на основе рекурсивных моделей анализа данных // Системная инженерия и информационные технологии. 2022. Т. 4. № 1 (8). С. 27—36. DOI: 10.54708/26585014_2022_41827.

23. **Miskinis C.** Disrupting the financial and banking services using digital twins. 2021. URL: https://www.challenge.org/insights/digital-twin-for-finance/

24. **Kümpel M., Mueller C., Beetz M.** Semantic Digital Twins for Retail Logistics // Dynamics in logistic. 2021. P. 129—153. DOI: 10.1007/978-3-030-88662-2_7.

25. **Matyi H., Tamás P.** Digital Twin Technology in Logistics Literature Review // Cutting & Tools in Technological System. December 2021. P. 13—21. DOI: 10.20998/2078-7405.2021.95.02.

26. **Ivanov S., Nikolskaia K., Radchenko G.** et al. Digital Twin of City: Concept Overview//In conference proceedings — 2020 Global Smart Industry Conference (GloSIC). 2020. P. 178—186. DOI: 10.1109/GloSIC50886.2020.9267879.

27. **Dembski F., Wössner U., Letzgus M.** et al. Urban Digital Twins for Smart Cities and Citizens: The Case Study of Herrenberg, Germany // Sustainability. 2020. Vol. 12, No. 6. Article 2307. DOI: 10.3390/su12062307.

28. **Lucas A.** Ising formulations of many NP problems // Frontiers in physics. 2014. 5 p. DOI: 10.3389/fphy.2014.00005.

29. **Nembrini R., Ferrari D. M. Cremonesi P.** Feature Selection for Recommender Systems with Quantum Computing // Entropy. 2021. Vol. 23, No. 8. Article 970. DOI: 10.3390/e23080970.

30. **Giordani M., Polese M., Mezzavilla M.** et al. Toward 6G networks: Use cases and technologies // IEEE Commun. Mag. 2020. Vol. 58. P. 55—61. DOI: 10.1109/MCOM.001.1900411.

31. **Allawi Y. M., Mohammed A. F. Y., Lee J., Choi S. G.** A Sustainable Business Model for a Neutral Host Supporting 5G and beyond (5GB) Ultra-Dense Networks: Challenges, Directions, and Architecture // Sensors. 2022. Vol. 22, No. 14. Article 5215. DOI: 10.3390/s22145215.

32. **Papidas A. G., Polyzos G. C.** Self-Organizing Networks for 5G and Beyond: A View from the Top // Future Internet. 2022. Vol. 14, No. 3. Article 95. DOI: 10.3390/fi14030095.

33. **Martínez-García A. N.** Artificial Intelligence for Sustainable Complex Socio-Technical-Economic Ecosystems // Computation. 2022. Vol. 10, No. 6. Article 95. DOI: 10.3390/computation10060095.

34. **Kim D., Jo D.** Effects on Co-Presence of a Virtual Human: A Comparison of Display and Interaction Types // Electronics. 2022. Vol. 11, No. 3. Article 367. DOI: 10.3390/electronics11030367.

35. **Инкубатор** интенсивной терапии новорожденных ИДН-03. Руководство по эксплуатации. Электронный ресурс. URL: https://www.xn--g1ajft.xn--p1ai/ru/production/medicina/neonatalnaya/idn-03

# System Engineering of the Organizational and Technical Systems' Digital Twins Using Artificial Intelligence Methods

**E. V. Orlova,** ekorl@mail.ru, Department of economics and management,
Ufa state aviation technical university, Ufa, 450008, Russian Federation

*Corresponding author:*

**Ekaterina V. Orlova,** D. Sc., Professor, Department of economics and management,
Ufa State Aviation Technical University, Ufa, 450008, Russian Federation
E-mail: ekorl@mail.ru

*Received on July 15, 2022*
*Accepted on August 18, 2022*

*The article considers the problem of digital twin design of organizational and technical systems. The theoretical and methodological basis of the research is the fundamental scientific works and applied works of Russian and foreign scientists in the field of digitalization and digital twins. Following methods are used in the research: system analysis, statistical analysis, operational research, artificial intelligence.*

*A comprehensive analysis of approaches and methods for digital twin design of organizational and technical systems is carried out. It is shown that for complex organizational and technical systems operating under uncertainty, there is no comprehensive, universal technology (methodological approach) for organization the process of developing digital twins in order to accelerate their engineering.*

*The technology for organizing a digital twin design, combining design stages, methods and models, and providing its system engineering is proposed.*

*The decision support model is developed for diagnosing the technical state of an object (technical device), based on the methods of situational analysis and fuzzy logic, which provides the synthesis of effective solutions under various situations and combinations of heterogeneous factors characterizing the object and its external environment.*

*The practical significance of the research is the developed decision support system, which is the basis for control system in solving problems related to monitoring the current state of technical devices (instruments, equipment) and developing adequate decisions to eliminate their disfunctions.*

***Keywords:*** *digital twin; methods for modeling digital twins; system design of digital twins; artificial intelligence methods*

*For citation:*

**Orlova E. V.** System Engineering of the Organizational and Technical Systems' Digital Twins Using Artificial Intelligence Methods, *Programmnaya Ingeneria,* 2022, vol. 13, no. 9, pp. 425—439.

DOI: 10.17587/prin.13.425-439

## References

1. **Prokhorov A., Lysachev M., Borovkov A.** *Digital twin. Analysis, trends, world experience,* Moscow, LLC "AlliancePrint", 2020, 401 p. (in Russian).

2. **Petrov A. V.** Simulation modeling as a basis for digital twin technology, *Bulletin of the Irkutsk State Technical University*, 2018, vol. 22, no. 10, pp. 56—66. DOI: 10.21285/1814-3520-2018-10-56-66 (in Russian).

3. **Qi Q., Tao F., Hu T.** et al. Enabling technologies and tools for digital twin, *Journal of Manufacturing Systems*, 2021, vol. 58, pp. 3—21. DOI: 10.1016/j.jmsy.2019.10.001.

4. **Orlova E. V.** Methodology and Models for Individuals' Creditworthiness Management Using Digital Footprint Data and Machine Learning Method, *Mathematics*, 2021, vol. 9, no. 15, 28 p. DOI: 10.3390/math9151820.

5. **Orlova E. V.** Innovation in Company Labor Productivity Management: Data Science Methods Application, *Applied System Innovation*, 2021, vol. 4, no. 3: 68. 18 p. DOI: 10.3390/ asi4030068.

6. **Banerjee P.** AI and ML: The Brave New World of Simulation, 2021, available at: https://www.ansys.com/blog/ai-and-ml-the-brave-new-world-of-simulation

7. **Radanliev P., De Roure D., Nicolescu R.** et al. Digital twins: artificial intelligence and the IoT cyber-physical systems in Industry 4.0, *International Journal of Intelligent Robotics and Applications,* 2022, vol. 6, no. 1, pp. 171—185. DOI: 10.1007/s41315-021-00180-5.

8. **Orlova E. V.** Model for Operational Optimal Control of Financial Recourses Distribution in a Company, *Computer Research and Modeling*, 2019, vol. 11, no. 2, pp. 343—358. DOI: 10.20537/2076-7633-2019-11-2-343-358 (in Russian).

9. **Orlova E. V.** Engineering of System Synthesis for Innovative Projects Efficiency, *Programmnaya Ingeneria,* 2019, vol. 10, no. 11—12, pp. 430—439. DOI: 10.17587/prin.10.430-439 (in Russian).

10. **Orlova E. V.** Methods and Models of Data Analysis and Machine Learning in the Problem of Labor Productivity Management, *Programmnaya Ingeneria*, 2020, vol. 11, no. 4, pp. 219—229. 10.17587/prin.11.219-229 (in Russian).

11. **Malik A. A., Masood T., Bilberg A.** Virtual reality in manufacturing: Immersive and collaborative artificial-reality in design of human-robot workspace, *International Journal of Computer Integrated Manufacturing*, 2019, vol. 33, no. 1, pp. 22—37. DOI: 10.1080/0951192X.2019.1690685.

12. **Tao F., Qi, Q.,Wang L., Nee A.** Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison, *Engineering,* 2019, vol. 5, no. 4, pp. 653—661. DOI: 10.1016/j.eng.2019.01.014.

13. **Mandolla C., Petruzzelli A., Percoco G., Urbinati A.** Building a Digital Twin for Additive Manufacturing through the Exploitation of Blockchain: A case analysis of the aircraft industry, *Computers in Industry,* 2019, vol. 109, pp. 134—152. DOI: 10.1016/j.compind.2019.04.011.

14. **Korostelkin A. A., Klyavin O. I., Aleshin M. V.** Optimization of Frame Mass in Crash Testing of Off — Road Vehicles*, Russian Engineering Research*, 2019, vol. 39, no. 12, pp. 1021—1028, at: https://link.springer.com/article/10.3103/S1068798X19120116

15. **Fonseca Í., Gaspar H., Mello P., Sasaki H.** A Standards-Based Digital Twin of an Experiment with a Scale Model Ship, *Computer-Aided Design*, 2022, vol. 145, article 103191. DOI: 10.1016/j.cad.2021.103191.

16. **Siebert T., Hack E., Labeas G., Patterson E., Splitthof K.** Uncertainty Quantification for DIC Displacement Measurements in Industrial Environments, *Experimental Techniques*, 2021, vol. 45, no. 5, pp. 685—694. DOI: 10.1007/s40799-021-00447-3.

17. **Future** role of digital twins in the aerospace industry, January 2019, available at: https://www.challenge.org/insights/digital-twin-in-aerospace/

18. **Patterson E., Diamantakos I., Dvurecenska K.** et al. Validation of a structural model of an aircraft cockpit panel: An industrial case study, *The Journal of Strain Analysis for Engineering Design*, 2021, 030932472110590. DOI: 10.1177/03093247211059084.

19. **Zborowski M.** Finding Meaning, Application for the Much-Discussed "Digital Twin", *Journal of Petroleum Technology*, 2018, vol. 70, pp. 26—32. DOI: 10.2118/0618-0026-JPT.

20. **Digital** Twins vs. Building Information Modeling (BIM), available at: https://www.iotforall.com/digital-twin-vs-bim/

21. **Bruynseels K., Santoni de Sio F., Van Den Hoven J.** Digital Twins in Health Care: Ethical Implications of an Emerging Engineering Paradigm, *Frontiers in Genetics*, 2018, vol. 9. DOI: 10.3389/fgene.2018.00031.

22. **Nasyrov R. V.** Causal approach to the construction of bionic calculations based on recursive data analysis models, *System Engineering and Information Technologies,* 2022, vol. 4, no. 1 (8), pp. 27—36. DOI 10.54708/26585014_2022_41827.

23. **Miskinis C.** Disrupting the financial and banking services using digital twins. 2021, available at: https://www.challenge.org/insights/digital-twin-for-finance/

24. **Kümpel M., Mueller C., Beetz M.** Semantic Digital Twins for Retail Logistics, *Dynamics in logistic,* 2021, pp. 129—153. DOI: 10.1007/978-3-030-88662-2_7.

25. **Matyi H., Tamás P.** Digital Twin Technology in Logistics Literature Review, *Cutting & Tools in Technological System*, December 2021, pp. 13—21. DOI: 10.20998/2078-7405.2021.95.02.

26. **Ivanov S., Nikolskaia K., Radchenko G.** et al. Digital Twin of City: Concept Overview, *In conference proceedings — 2020 Global Smart Industry Conference (GloSIC),* 2020, pp. 178—186. DOI: 10.1109/GloSIC50886.2020.9267879.

27. **Dembski F., Wössner U., Letzgus M.** et al. Urban Digital Twins for Smart Cities and Citizens: The Case Study of Herrenberg, Germany, *Sustainability*, 2020, vol. 12, no. 6, article 2307. DOI: 10.3390/su12062307.

28. **Lucas A.** Ising formulations of many NP problems, *Frontiers in physics*, 2014, 5 p. DOI: 10.3389/fphy.2014.00005.

29. **Nembrini R., Ferrari D. M. Cremonesi P.** Feature Selection for Recommender Systems with Quantum Computing, *Entropy,* 2021, vol. 23, no. 8, article 970. DOI: 10.3390/e23080970.

30. **Giordani M., Polese M., Mezzavilla M.** et al. Toward 6G networks: Use cases and technologies, *IEEE Commun. Mag.,* 2020, vol. 58, pp. 55—61. DOI: 10.1109/MCOM.001.1900411.

31. **Allawi Y. M., Mohammed A. F. Y., Lee J., Choi S. G.** A Sustainable Business Model for a Neutral Host Supporting 5G and beyond (5GB) Ultra-Dense Networks: Challenges, Directions, and Architecture, *Sensors*, 2022, vol. 22, no. 14, article 5215. DOI: 10.3390/s22145215.

32. **Papidas A. G., Polyzos G. C.** Self-Organizing Networks for 5G and Beyond: A View from the Top, *Future Internet,* 2022, vol. 14, no. 3, article 95. DOI: 10.3390/fi14030095.

33. **Martínez-García A. N.** Artificial Intelligence for Sustainable Complex Socio-Technical-Economic Ecosystems, *Computation,* 2022, vol. 10, no. 6, article 95. DOI: 10.3390/computation10060095.

34. **Kim D., Jo D.** Effects on Co-Presence of a Virtual Human: A Comparison of Display and Interaction Types, *Electronics*, 2022, vol. 11, no. 3, article 367. DOI: 10.3390/electronics11030367.

35. **Intensive** care incubator for newborns IDN-03. Operation manual, available at: https://www.xn--g1ajft.xn--p1ai/en/production/medicina/neonatalnaya/idn-03

DOI: 10.17587/prin.13.440-448

# Self-Synthesis of Programs Based on Artificial Chemistry Model

**E. A. Kol'chugina,** kea_sci@list.ru, Department of Mathematical Support and Computer Application, Penza State University, Penza, 440026, Russian Federation

*Corresponding author:*

**Elena A. Kol'chugina,** D. Sci., Professor, Department of Mathematical Support and Computer Application, Penza State University, Penza, 440026, Russian Federation
E-mail: kea_sci@list.ru

*Received on May 15, 2022*
*Accepted on September 05, 2022*

*Fully automatic software synthesis will become possible when simpler programs or program components become able to spontaneously and inevitably attract each other and connect with each other. To achieve this, it is necessary to construct special means to provide spontaneous interactions between programs, since currently there are no such means. In this article, using the principles of artificial chemistry, we propose an algebra of "sinks-and-sources", a concept of artificial atom and a model called $H_2O$. We also describe experiments with this model, in which the formation of a model molecule of "water" is reproduced. We represent artificial atoms of simple substances, such as oxygen and hydrogen, as independent processes corresponding to two types of programs. We construct connections between individual atoms using sockets, which are necessary to simulate shared particles from outer orbitals. During the experiments, we registered the emergence of artificial molecules of "water" and other complex substances. The experimental results prove that simple software units implementing the proposed principles are capable of spontaneously forming complex software structures without directed external influence. The achieved results are useful to software engineering, artificial life and artificial intelligence to provide self-development of software and complex logical structures capable of further evolution and self-improvement.*

***Keywords:*** *automatic software synthesis, spontaneous self-formation of complex programs, artificial chemistry, artificial atom*

*For citation:*

**Kol'chugina E. A.** Self-Synthesis of Programs Based on Artificial Chemistry Model, *Programmnaya Ingeneria,* 2022, vol. 13, no. 9, pp. 440—448.

УДК 004.42

**Е. А. Кольчугина,** д-р техн. наук, доц., проф. кафедры, kea_sci@list.ru,
ФГБОУ ВО "Пензенский государственный университет"

# Самосинтез программ на основе модели искусственной химии

*Полностью автоматический синтез программного обеспечения станет возможным, когда более простые программы или программные компоненты будут способны спонтанно и неотвратимо притягивать друг друга и соединяться. Для этой цели необходимо сконструировать специальные средства обеспечения спонтанного взаимодействия между программами, поскольку в настоящее время таких средств нет. В статье на основе принципов искусственной химии предложены алгебра "стоков" и "источников", концепция искусственного атома и модель под названием $H_2O$. Также описаны эксперименты с этой моделью, в которых воспроизводится*

*образование модельной молекулы "воды". Искусственные атомы простых веществ, таких как кислород и водород, представлены как независимые процессы, соответствующие двум типам программ. Соединения между отдельными атомами строятся с помощью сокетов, которые необходимы для имитации общих частиц с внешних орбиталей. В ходе экспериментов было зарегистрировано появление искусственных молекул "воды" и других сложных веществ. Экспериментальные результаты доказали, что простые программные единицы, реализующие предложенные принципы, способны самопроизвольно формировать сложные программные структуры без направленного внешнего воздействия. Достигнутые результаты полезны для программной инженерии, искусственной жизни и искусственного интеллекта для саморазвития программного обеспечения и сложных логических структур, способных к дальнейшей эволюции и самосовершенствованию.*

***Ключевые слова:*** *автоматический синтез программного обеспечения, спонтанное самообразование сложных программ, искусственная химия, искусственный атом*

## Introduction and background

Software is an important component of modern technologies, having a critical impact on their quality and efficiency. Therefore, the enhancement of software, its paradigms and principles of development, in turn, is important. The reality of the modern software development process has two significant aspects.

Firstly, this is an aspect related to the organization and technologies of software development. Modern software is created by a team of developers who have different specializations of the programmer profession and often do not interact directly with each other. Sometimes the complexity and scale of development is such that the project cannot be fully comprehend and controlled by one person. This complicates the software development process and reduces its reliability due to influence of the limitations based on the human factor.

Secondly, there is an aspect concerning the life cycle of software. Programs tend to become obsolete quickly, and during exploitation they also need to be improved and adapted. The amount of work on the revision and adaptation of the software may be insignificant, but these works require the involvement of specialists and time-consuming. This means spending additional resources to create and maintain the project, which reduces its effectiveness.

One of the possible solutions is to create programming paradigms and technologies that allow to automate the development, improvement and generation of new versions. This process should take place under supervision, but without the constant direct participation of a specialist.

In this article, we propose a mechanism that can provide spontaneous self-development of programs under certain conditions. This mechanism is designed to ensure the formation of program structures based on independent units endowed with specific means of interaction.

The scope of application of the proposed mechanism for self-development of programs can be arbitrary, regardless of the application. The proposed mechanism is intended for the development of general-purpose software and is compatible with imperative programming tools, allowing the use of such popular languages as C/C++, Pascal, Perl.

Chemistry serves as a metaphor for creating such a mechanism. Programs should be formed as a result of interactions with other programs, similar to the formation of substances in chemistry as a result of reactions. Among computer models, the closest analogies are models of artificial chemistry.

In artificial chemistry models [1—3], programs are considered as artificial molecules capable to react with each other. As a result of the reactions, the molecules (i. e. programs) can combine to form larger and more complex molecules or programs. So, using the terminology of chemistry, the essence of solving the problem of self-development of programs is to provide conditions for the spontaneous synthesis of complex artificial molecules from simpler ones.

To achieve this, we must possess means to artificially reproduce forces of attraction and repulsion to make self-synthesis of artificial molecules selective and asymmetric. We also need to endow our artificial molecules with energy. The standard analogue of energy in artificial life and artificial chemistry models [2, 3] is central processor time. To get energy, artificial molecules, or programs, must be activated as executing processes. To retain energy, the processes should strive to extend their execution as long as it possible. A successful strategy for obtaining such a result is the formation of cyclic structures, as in living systems.

Life is an existence in a form of dynamically stable cyclic repetition of the same set of physiology processes or, in simpler case, reactions. Such an existence possible as long as energy needed to carry out the processes (reactions) is available. That form of existence observed both in natural biological life [4] and in supposed non-protein forms of living, e. g. based on nuclear reactions occurring inside the stars [5], or on computations in artificial life models [3, 6—8].

In models of artificial life and artificial chemistry, cyclic structures representing sets of coupled reactions or artificial analogues of molecular organisms must earn demanded amount of energy by performing specified calculations. This simultaneously prolongs the existence of structures and makes them capable of evolutionary development.

The model of artificial life proposed in [8] was aimed at studying the emergence of such cyclic computing structures. In the experiments described in [8], it was shown that in distributed systems, cyclic structures can sponta-

neously arise, which manifest themselves in the form of repetitive executions of indirectly interacting processes (functions).

These structures arising in presence of shared variables, which are necessary as input data for processes. But the arising structures are fragile. The cycles are easily corrupted or completely destroyed by competing processes that avariciously searching for the same input data.

The problem outlined in [3, 8] is that there are no means to adequately represent the forces of repulsion and attraction in models, on the one hand, and the selectivity and asymmetry of interactions, on the other.

Let us consider and compare the known methods of representing relations and interactions, more precisely, forces of attraction and repulsion. Studying the literary sources concerning models of artificial life and artificial chemistry, we can find at least three types of such methods:

- based on adjacency and proximity in cellular automata space, e. g. [9, 10];
- based on the use of common objects, primarily data structures, e.g [3, 8];
- based on the rewriting and composition of functions, e. g. [11].

The oldest and most obvious method of representing the forces of attraction came from the theory of cellular automata [9, l0]. This method is based on the proximity and neighbourhood factors. The closer the neighbour is to the cell, the stronger its influence on the cell. The most influential cells are adjacent cells: they can participate in mutual state transformations (reactions) according to the transition rules. The big disadvantage of this method is the need for centralized management to choose which of the possible transition rules should be preferred, or more precisely, which sequence of rule execution should be performed. This management is performed by central processor unit (CPU). Thus, execution of the rules is not arbitrary, although in general selection of rules is equiprobable, and the strength and asymmetry of the rules are not defined. The repulsion force is not represented at all. But, as it was stated earlier, it is preferable that interactions occur spontaneously, due to the properties inherent in the interacting analogues of the molecules, that is, the programs. Thus, the use of proximity and neighbour factors is not quite suitable.

The second group of methods for representing and establishing relationships is based on the use of common objects: data structures or labels. The overview and analysis of such methods is given, for example, in [3]. But again we are faced with the same problem: the need for a decision-making centre that must authorize interaction and supply it with a model analogue of energy. The only energy supply centre is, as it was stated before, the CPU. In [3, 8], it was revealed that the object methods of representing relations are weak, indirect, and hardly adequate. They also suffer from a lack of selectivity and asymmetry. And again, there is no explicit representation of the repulsive force.

The third way to represent attraction between programs, which are understood as artificial molecules, is based on rewriting their source codes and constructing compositions. In [11], the interaction between programs that leads to formation of their composition is initiated by the decision-making centre. This centre arbitrary chooses interacting programs, thereby representing attraction. The repulsive force is not represented.

Let's formulate and clarify the requirements necessary to obtain the desired representation of attraction-repulsion relations in computer models:

- these relationships should occur spontaneously and inevitably;
- relations should be represented through dynamic interactions, not entities;
- a model in general should not have a decision-making centre, an arbitrator, or an initiating energy centre;
- emerging relationships should be direct and stable;
- selectivity and asymmetry are also necessary.

Based on this list of requirements, we return now to the analysis of the model from [8] and its shortcomings, and also determine ways to improve the model.

According to one of our statements, any relation must be represented as a form of dynamics that results from the activity of the interacting parts. The model from [8] was proposed mainly to achieve this. The purpose of this model was to study the possibility of the emergence of dynamic cyclic structures formed from a set of independently functioning processes. These processes unintentionally interacted with each other by transforming data (symbols) stored in separate spatial cells.

All in all, this method is close to the methods of the second group described above, but there are some small but important features. There is no decision-making centre in this model, and it is unpredictable which process will access the data cell first. Also, in this model, there are no explicitly defined "gluing" common objects, such objects appear unintentionally.

These features allow studying self-organization of connections between unspecific common-purpose application programs. This is very important for software development research. But the resulting connections are weak, and the programs in the structures are loosely coupled. The repulsive force is represented implicitly, as a consequence of the inability of the process to establish a connection, which compels the process to explore other spatial cells.

Thus, as indicated in [3], we always face the same set of problems: the lack of spontaneity; the lack of representation of the concepts of force and electric charge; the inadequate representation of the interactions of attraction and repulsion; the inability to represent the asymmetry and selectivity of the interactions. The lack of selectivity and asymmetry implies that two independent processes implementing different functions, for example, $f$ and $g$, consuming the same data $A$, have an equal ability to be attracted to the cell containing the data, and there is no means to change this.

To overcome this set of problems, in [12] it was proposed to establish the principles of artificial physics, a new scientific branch for the study of principles and methods that allow reflecting the principles and categories of real-world physics in computer models. By computer models,

we mean, first of all, models of artificial life and artificial chemistry.

Also in [12], a method for representing the electric charge was proposed. The non-specificity of programs in model from [8] is both a strong and a weak side. Previously, we outlined the strong side: it is possible to study unintentional formation of structures from the set of randomly chosen applications. But the non-specificity makes connections weak and hinders selectivity. To eliminate this, we must sacrifice the non-specificity and endow programs with "hooks and loops" that allow attraction modelling. This corresponds to the "key-lock" or "induced fit" principles in the real-world chemistry [13].

Thus, programs or artificial molecules should get specific tools that allow them to "hook" another program or "be hooked" by another program. These tools are intended to mimic the action of an electric charge or the interaction of elementary particles in atoms and molecules of the real-world physics and chemistry. Most importantly, the relations between programs should not be represented by tools that are understood as common objects, but by interactions that are performed using such tools. In the real-world chemistry, the chemical composition of substances determines not only the properties of substances, but also the set of possible reactions and their course. Similarly, tools that model "hooks" and "loops" should determine the possible interactions of artificial molecules, i. e. programs.

In [12], it was suggested to use sockets as such tools. These tools are communication tools that enable interactions between processes. Therefore, they are the means to establish relations in the form of dynamics. This is a unique feature of sockets as inter-process communication system (IPC) tools.

According to [12], the "hooks and loops" enabling attraction are understood as "sinks" and "sources" that jointly model an electric charge. The "sources" represent electrons from the outer orbitals that are loosely bound to the nucleus and can be attracted by other atoms or ions. The "sinks" represent free positions in the outer orbitals that can be occupied by additional electrons attracted from other atoms or ions.

Further in this article, we consider a model illustrating formation of bonds between artificial molecules, or programs, which for the simplicity are understood as monatomic molecules. As a result, this will allow subsequent switching to a more adequate understanding of a program as an analogue of a polymer molecule, or more correctly, an enzyme with several centres of polarization.

## 1. Method

### 1.1. General concept

A model concept is a quintuple of the following type:

$$V = \langle S, T, A, G, M \rangle,$$

where $S$ — model space; $T$ — model time; $A$ — a set of atoms (monatomic molecules) or symbols in alphabet; $G$ — a set of rules (grammar rules) to produce complex molecules (words, or symbol chains) from monatomic molecules or atoms (i. e. letters, symbols); $M$ — a set of all permitted molecule types (a vocabulary of all possible words).

As in [8], all simulation takes place in two-dimensional closed cellular space $S \subseteq (\mathbf{N}_{0+} \times \mathbf{N}_{0+})$, $\mathbf{N}_{0+}$ here and further is a set of all non-negative integer numbers. The model time, like in [8], is linear unidirectional: $T \subseteq \mathbf{N}_{0+}$.

The full definition of atoms, or monatomic molecules, is

$$f = \begin{cases} \varnothing, \\ f() = f, \\ f(z_1, \ldots, z_n),\ n \in \{1, \ldots, \infty\}. \end{cases}$$

This means that the function can be empty, with null arguments, or with single or multiple arguments.

In general case, a monatomic molecule $f \in A$ is a multiargument function $f(x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_m)$, where $\mathbf{X} = \langle x_1, x_2, \ldots, x_n \rangle$ is a vector of variables or null-argument functions, $\mathbf{Y} = \langle y_1, y_2, \ldots, y_m \rangle$ is a sequence of slot symbols, $\forall y_i \in \{>, <\}$. This slot symbols represent "sinks" (<) and "sources" (>) necessary to simulate electric charge and attraction-repelling force. Any slot symbol indicates a position where another function could be inserted to create composite function.

The slot symbols provide formation of the complex molecules according to the following rules from $G$.

1. First rule is an attraction rule. Slots symbols act in the reciprocal way: a function $f(X_1, >, Y_1)$ could be inserted in a function $g(X_2, <, Y_2)$ instead of symbol <, or vice versa. The resulting transformation is written as $f(X_1, > g(X_2, Y_2) <, Y_1)$ or $g(X_2, < f(X_1, Y_1) >, Y_2)$, both forms are mutually dual and technically describe the same thing. Note that both forms $f(X_1, > g(X_2, Y_2) <, Y_1)$ and $g(X_2, < f(X_1, Y_1) >, Y_2)$ represent only data connections between functions, not the order in which they are calculated. In general, the functions are calculated concurrently.

2. The second rule, or the rule of repulsion, establishes that compositions between functions with similar slot symbols, e. g. $f(X_1, >, Y_1)$ and $g(X_2, >, Y_2)$, or $f(X_1, <, Y_1)$ and $g(X_2, <, Y_2)$, are prohibited.

3. The third rule defines the decomposition method. Any complex molecule $f(X_1, > g(X_2, Y_2) <, Y_1)$ or $g(X_2, < f(X_1, Y_1) >, Y_2)$ may spontaneously decay into $f(X_1, >, Y_1)$ and $g(X_2, <, Y_2)$. The direction of the brackets in complex molecules provides the determination of the corresponding slot symbols for the components of decay.

Let us add a definition of the strength of interactions into our model. The strength of interactions is represented by number of slot symbols. The number of given slot symbols can be "neutralized" by the same number of oppositely directed symbols. This means that $n$ slot symbols in function $f(X_1, >_1 \ldots >_n, Y_1)$ can attract $1 \leqslant p \leqslant n$ functions with oppositely directed slot symbols. In the extreme case, this could be done by

single function with the same strength of attraction: $g(X_2, <_1 \ldots <_n, Y_2)$. Such attraction creates mutually dual formulas like $f(X_1, >_1 \ldots >_n g(X_2, Y_2) <_n \ldots <_1, Y_1)$ and $g(X_2, <_1 \ldots <_n f(X_1, Y_1) >_n \ldots >_1, Y_2)$. Another extreme case occurs when $f(X_1, >_1 \ldots >_n, Y_1)$ attracts $n$ various independent functions like $h_1(X_{11}, <, Y_{11}), \ldots, h_n(X_{1n}, <, Y_{1n})$, each of which has a single slot symbol <. The resulting complex molecule is $f(X_1, < h_1(X_{11}, <, Y_{11}) >, \ldots, < h_n(X_{1n}, <, Y_{1n}) >, Y_1)$.

A set of all complex molecules composed from $A$ by applying the rules from $G$ is $M$. The atoms and molecules from $A$ and $M$ respectively are distributed over the model space, and this distribution is changing in time. Hence, the evolution of our artificial chemistry is a mapping $S \times T \to M \cup A$.

Note that the considered resulting complex molecules have a linear or tree-like structure. It is possible to form more complex and universal structures, for example, cyclic ones, but these cases are not considered here.

Cyclic structures can be easily obtained by slightly enhancing this model. However, we do not consider this enhancement here, so as not to exceed the volume of the article and not distract from priority task, which is the reproduction of the forces of attraction and repulsion like those that act between charged particles. Our next work will be devoted solely to the representation of cyclic structures. Now we focus our attention on the general principle of the emergence of dynamic structures.

In the quintuple $V = \langle S, T, A, G, M \rangle$, nearly all components of artificial chemistry are present: $A \cup M$ is a set of all possible molecules; $G$ is a set of rules; $\langle S, T \rangle$ represents the space-time structure of reaction vessel or domain, and the only thing that needs to be added to complete the definition of artificial chemistry is an algorithm for applying rules from $G$. The aims of this work are to achieve spontaneity of interactions and avoid introducing a decision centre into the model, so we are to assume that the rules are applied arbitrarily. The described model is naturally suitable for parallel systems with a fine-grained structure, where each atom or molecule has its own CPU. In this case, we are dealing with a multitude of independent decision-making centres that possesses their own energy. This makes the process of interaction between atoms and molecules stochastic. Thus, in our artificial chemistry, the algorithm for applying rules from $G$ is an empty set of operators.

Practically, the triple $\langle A, G, M \rangle$ is a grammar, but it also defines an algebra of "sinks" and "sources". This abstract algebra of "sinks-and-sources" is the basis for a more specific computational model that describes the process of forming complex programs from simpler ones. This model is considered and discussed below.

### 1.2. $H_2O$ model

Here we present a computer model to reproduce the formation of a water molecule from simpler substances such as oxygen and hydrogen. That is why we name this model $H_2O$. The model is obtained by instantiating our base model $V$.

As it was mentioned above, the model space $S$ in $H_2O$ model is cellular and two-dimensional. Any cell of the model space $S$ can be empty or occupied by a single atom or molecule. The atoms composing the set $A$ belong to hydrogen-like or oxygen-like types. Molecules form $M$ can be monatomic, diatomic, or more complex.

According to definition of the model $V$ and the algebra of "sinks-and-sources", it is possible to formulate the concept of an artificial atom as a dynamic entity performing a computational function, possessing an analogue of energy (CPU time) and capable of interacting with other similar entities through "sinks" or "sources" represented by slots in the notation of $V$. Thus, an abstract artificial atom is a combination of a function, the amount of CPU time and a set of slots.

A hydrogen-like atom (hereinafter, for simplicity, "hydrogen" atom) has a single electron in its outer orbit, which can be shared with other artificial atoms of the same or a different kind. Thus, usually it represents "source" of electric charge. To make our model more intricate and interesting, we consider the outer orbital of "hydrogen" atom as not completely filled. There is an empty position there. So, the "hydrogen" atom is both a "sink" and a "source" of electric charge at the same time. In the proposed model, a "hydrogen" atom or a monatomic molecule is light and capable of migration.

An oxygen-like atom (hereinafter, for simplicity, "oxygen" atom) needs two additional electrons to fill the outer orbitals. This atom has a simpler functionality than "hydrogen" and represents only "sink" of charge. Such artificial monatomic molecule is heavier than "hydrogen" and occupies the spatial cell motionlessly.

In addition, as it was proposed in [8], the emptiness of the cell can be considered as a special type of atom, an analogue of an inert gas that is unable to form compounds with other atoms. So we add a special type of atoms $\varnothing$ to $A$. These atoms perform an empty function; they do not have CPU time portions and slots.

Atoms and molecules can establish relationships provided that they are in the same spatial cell at the same time. This allows atoms and molecules to find a suitable partner by looking for a logical channel or port corresponding to the socket. The connection between atoms or molecules established through a socket can subsequently be broken by any of the interacting partners at any time, or, on the contrary, last indefinitely.

It is obvious that our model combines the characteristics of at least two methods of representing relationships and interactions that were identified earlier: to establish relationships between molecules and atoms, the proximity of reacting partners is necessary; on the other hand, reacting atoms and molecules must have something in common and joint, i. e. a common virtual particle or a model electron.

It is known that in the client-server architecture, the client part plays an active role in establishing interconnections, while the server part is passive and driven. According to this, we consider a negative charge as active

and represent it as a client socket. The excess positive charge is represented in the model as a passive and recipient server socket.

The port number plays the role of a principle quantum number that identifies an electron shell or a set of electron orbitals with the same energy level. To establish a connection between artificial atoms, an artificial atom possessing free electrons tries to share them with the other model atoms located in the same spatial cell. In the model $H_2O$ this corresponds to polling of the ports associated with sockets. When searching for a port, the "first response" rule is applied: the number of the first responding port is used.

The port number is associated with both the location, i. e. the coordinates of the spatial cell, and the number of the represented period (orbit) according to the periodic table of artificial elements.

The active client side polls the port numbers in descending order, that is, from the outer orbitals to the inner ones. Therefore, the construction of a model "water" molecule is more probable than the construction of a model diatomic "hydrogen" molecule $H_2$.

Possible chemical compounds consisting of artificial "hydrogen" and "oxygen" are shown in fig. 1. At the same time, fig. 1 represents graphic language capable to describe combinations of artificial molecules according "sinks-and-sources" algebra. An atom is represented by circle endowed with "sinks" and "sources", which allow to establish connections with other atoms through their "sources" and "sinks" respectively.

Figure 1 demonstrates the tendency present in our artificial model chemistry to form long polymer-like molecules containing chains of "hydrogen" atoms and "hydroxide" groups. Some of these molecules are exotic and do not exist in nature, but in our artificial simplified chemistry, their existence is quite likely.

In our model, we did not aim to fully reproduce the chemistry of water and hydrogen compounds. We are only interested in the process of formation and dynamic transformation of structures, so the compounds in our model are intentionally unstable, and the "hydrogen" atoms are easily detaching and highly volatile. This approach makes it possible to obtain and track the history of many compounds, unlike chemistry with the formation of strong stable bonds. At this stage of research, it is more important to ensure the ability of "atoms" to regroup. If necessary, the stability of the connections can be achieved by slightly changing the parameters of the model.

To study dynamics occurring in the proposed artificial chemistry system, it was necessary to conduct an experiment described below.

## 2. Experiment description

To conduct an experiment with the previously proposed model $H_2O$, we created software in Perl. This software consists of several independent programs. The first and most important of them is a modeling tool that implements the previously described model. This main simulation program also generates an initial configuration file and maintains the system logs of events. The remaining programs are auxiliary. They are necessary for analyzing statistics and presenting it in a convenient and understandable form. The simulation begins with the execution of main program. When the main simulation program is terminated, auxiliary programs are started and configuration and log files are analyzed.

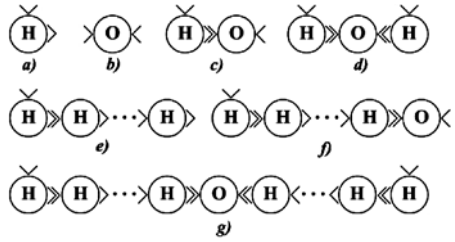


**Fig. 1. Artificial atoms and possible molecules in $H_2O$ model:**
*a* — "hydrogen"; *b* — "oxygen"; *c* — "hydroxide" OH; *d* — "water"; *e* — "polymeric hydrogen"; *f* — "polymeric hydrogen with hydroxide group"; *g* — analogues of hydronium type molecules, such as $H_1OH_n$ or $H_mOH_n$

The key difference between the main simulation program and our previous development described in [8] is that in this study, the independent processes are not associated with functions and substitution rules, but with artificial atoms. In the model $H_2O$, we also use sockets as the main means of interprocess communication instead of shared memory in [8].

At the beginning of any experiment, each of independent processes associated with artificial atom of "hydrogen" or "oxygen" goes through initialization phase. At this point, the process creates system of sockets, where passive server sockets are "sinks" and active client sockets are "sources".

Each process is also connected to a location in the cell model space, being associated to randomly assigned cell coordinates ($x$, $y$). The process representing "oxygen" is not mobile and is rigidly associated with its current location. On the contrary, the process representing volatile "hydrogen" is mobile and capable of exploring various spatial cells. The "hydrogen" process can change its position by observing its current location in search of molecules and atoms with which to react. Hence, "hydrogen" processes initially are active explorers, while "oxygen" processes are passive responders.

To create effective server socket, we must control the actual number of established and possible connections. To achieve this, we use a solution based on the pre-forking method from [14]. According to this method, the server must maintain a pool of child processes, and each of these processes is waiting for binding to an incoming request from the client. This method has at least two advantages.

First, it is easy to take into account and control both a quantity of child processes and their system identifiers. Knowledge of system identifiers allows controlling the execution and behaviour of child processes by sending commands to them. Knowing the number of executing child processes, it is possible to maintain a pool of ready child processes of constant size. Secondly, the availabil-

ity of ready child processes significantly increases the performance of the model by reducing the time expected for the creation of child processes. But the price for such advantages is the complication of the model.

The possibility of success of the experiment is based on the principle of port numbering:

$$port\_number = base + aton(concat(x, y, orbit, orbital)),$$

where $port\_number$ — the number of server port; $base$ — a starting point, an arbitrary chosen number; $x$, $y$ — the coordinates of the spatial cell; $orbit$ — the number of the corresponding atom orbit, $orbital$ — the number of the corresponding atom orbital; $aton$ — a function that converts a string into a number; $concat$ — the concatenation function.

Any port number can simultaneously receive as many incoming requests as there are currently vacant positions available in the corresponding orbit.

As it was previously stated, we used Perl to create modeling software and the main simulation program too. The problem with simulation in Perl is the measuring of time. For Perl programming, the natural and most convenient unit of measurement of time is the second. But the computation speed is much higher, and the same is true for the speed of movement of "hydrogen" atoms in the model, respectively. Therefore, it is necessary to slow down the movement of artificial atoms along the spatial lattice in order to make the measurement of time and speed of movement comparable and so that, as a result, the events of the formation of molecules become observable and distinguishable.

In the case of practical implementation, there is no need to perform such a slowdown. But in the case of an experiment, it is necessary to make sure that our initial assumptions are correct. Namely, that two atoms of "hydrogen" can be attached to one atom of "oxygen" simultaneously, and strictly no more than two.

One of the functions of the main simulation program is event logging. The program produces a set of such event logs. Each log is associated with a "sink" socket belonging to an "oxygen" atom.

Processes spawned by the main program and representing "sinks" implemented as server sockets record information about newly received and already available incoming connections (events) in their logs. This information of events is represented as a pair consisting of a time coordinate and the identifier of the connected client process (i. e., the "hydrogen" atom). In more complex cases, the identifier of the process modeling the "hydrogen" atom may be accompanied by a chain of identifiers of the processes associated with it.

The purpose of the auxiliary programs is to process and present statistical data. The first of these programs explores the temporal sequence of events in spatial cells. To do this, it generates files for each of the spatial cells, where all the events that occurred at a given point in space are indicated for each time coordinate. Owing to this ordering, it is possible to identify the formation of any complex molecule, for example, a "water" molecule when two "hydrogen" atoms join one "oxygen" atom. This approach allows us to study the simulation results from the point of view of the activity of "oxygen" atoms.

The second program examines statistics from the point of view of "hydrogen" activity. This program processes all log files and creates report files for each "hydrogen" atom present in the model. These reports reflect the trajectories of the mobile "hydrogen" atoms. Any record contains the identifier of the process simulating "hydrogen", the time coordinate and the coordinates of the spatial lattice.

In experiments, the size of spatial lattice varied from 4×4 to 8×8 cells. According a typical scheme, each of the model runs lasted for 30 min, and they were repeated many times with the same results.

The experiment consisted of two parts. In the first part of the experiment, the artificial atom of "hydrogen" implemented only the function of the "source". In the second part of the experiment, the artificial "hydrogen" atom, after joining the "oxygen" atom, also performed the function of a "sink".

In experiments, artificial atoms performed only the simplest operations on the data. The artificial atom of "oxygen" received messages coming through the "sinks" and placed the messages in the log. An artificial atom of "hydrogen" received messages through a "sink" (if there was one), added its own message and transmitted all of them further through a socket simulating a "source".

The results of the experiments are presented in fig. 2 and 3.

```
1648500413 - (2,3) - 19159
1648500418 - (2,3) - 19159
1648500433 - (2,2) - 19159
1648500448 - (1,2) - 19159
1648500473 - (2,3) - 19159
1648500478 - (2,3) - 19159
1648500483 - (0,1) - 19159
1648500488 - (0,1) - 19159
1648500493 - (0,1) - 19159
1648500498 - (0,1) - 19159
1648500503 - (0,1) - 19159 19152
1648500508 - (0,1) - 19159 19152
1648500523 - (2,0) - 19150 19159
1648500533 - (2,0) - 19159 19150
1648500538 - (2,0) - 19150 19159
1648500543 - (2,0) - 19154 19159
1648500548 - (2,0) - 19154 19159
1648500553 - (2,0) - 19154 19159
1648500558 - (2,0) - 19154 19159
1648500563 - (2,0) - 19159 19154
1648500568 - (2,0) - 19154 19159
1648500573 - (2,2) - 19159
1648500578 - (2,2) - 19159 19154
1648500583 - (3,3) - 19159
1648500588 - (3,1) - 19159
1648500593 - (3,1) - 19159 19154
```

**Fig. 2. A fragment of log registering movements of process with PID = 19159 (artificial "hydrogen" atom)**

```
1651251751 - (1,1) - <H-4397->
1651251753 - (1,1) - <H-4379-> <H-4397->
1651251755 - (1,1) - <H-4397-H-4437->
1651251757 - (1,1) - <H-4380-> <H-4397-H-4441->
1651251759 - (1,1) - <H-4380-H-4401-> <H-4397-H-4441->
1651251787 - (0,1) - <H-4397-> <H-4391->
1651251797 - (1,1) - <H-4397-> <H-4389->
1651251799 - (1,1) - <H-4397->
1651251801 - (1,1) - <H-4397-> <H-4405->
1651251803 - (0,0) - <H-4389-> <H-4397->
1651251803 - (1,0) - <H-4397->
1651251805 - (0,0) - <H-4397->
1651251805 - (1,0) - <H-4397-> <H-4378->
1651251807 - (0,0) - <H-4378-> <H-4397->
```

**Fig. 3. A fragment of log registering movements of process with PID = 4397 (a configuration with two "sleeves" is present)**

## 3. Results and discussion

An example of the results for the first part of experiment is shown in fig. 2. This figure shows a fragment of the event log. This event log illustrates the history of the movement of a single artificial atom of "hydrogen". This atom is modeled by a process with the identifier (PID) 19159. The first column represents a time coordinate; the second column represents spatial coordinates (pairs of numbers given in parentheses). The third column contains the PIDs of processes representing "hydrogen" atoms and attached to the same "oxygen" atom in the same spatial cell. One of the PIDs in the sequence is always equal to 19159, since the behavior of this particular process is being investigated.

This fragment of the log proves that spontaneous formation of "hydroxide" (fig. 1, *c*) and "water" (fig. 1, *d*) molecules is possible, and the molecules are quite stable. We see repetitive combinations of $\{19159, 19154\}$, $\{19159, 19150\}$ and $\{19159, 19152\}$. From 1648500503 to 1648500508, and from 1648500523 to 1648500568, there were "water" molecules in the cells (0,1) and (2,0), respectively. The chemical composition of these molecules is constant, although the atoms forming them change.

For the second part of experiment, we have modified the form of data output. The conditions of this part of the experiment allow each atom of "hydrogen" attached to an atom of "oxygen" to attract other atoms of "hydrogen". Consequently, the formation of the molecules shown in fig. 1, *f* and fig. 1, *g* is possible under such conditions.

The most complicated configuration of the molecule has become similar to that shown in fig. 1, *g*. This is a configuration consisting of two "sleeves" coming out of a common centre. Each "sleeve" is represented as a sequence of "*H*" symbols denoting "hydrogen" and the PIDs of the corresponding processes. In addition, any of these sequences are enclosed in angle brackets. A log fragment illustrating configurations involving a process with PID = 4397 is shown in fig. 3. It is obvious that at the moment of 1651251759, a configuration with two "sleeves", each of two molecules, has been registered.

Analyzing the results of the experiments, we should note that simple programs equipped with standard interaction mechanisms can spontaneously and unintentionally combine into complex structures. These structures are more obvious and strongly related than those observed in [8]. There are direct explicit connections between simple programs in such structures, unlike those observed in [8].

The attraction between reacting program components is obviously present. Asymmetry and selectivity of interactions are also present in the model, but the repulsive force is still implicitly represented. It is present as the impossibility of attraction.

All types of complex artificial molecules shown in fig. 1 can emerge in the model, with the exception of "polymer hydrogen" (fig. 1, *e*). The reason is the high mobility of "hydrogen" in the model.

In the model, there is one type of atoms that give up electrons ("hydrogen"), and two types of atoms that accept electrons (both "hydrogen" and "oxygen"). This creates an asymmetry towards the predominance of a positive charge. As a result, the artificial model "hydroxide" is obviously positively charged, whereas in nature it is an anion. Hydrogen bonds between molecules are not possible in the model, the obstacle is also related to the problem of charge representation.

In general, this model is only the first approximation on the way to creating artificial chemistry, in which developed program structures can be spontaneously formed from relatively independent components. The model as a whole and the artificial atom model need further improvement.

## Conclusions

In this article, we proposed a general concept of an artificial chemistry model capable of reproducing spontaneous interactions between individual programs (functions) performed by independent processes. This concept includes the algebra of "sinks-and-sources" that allow to describe the rules for the self-formation of programs. Compared with λ-calculus [15], this algebra allows forming programs with arbitrary structures.

The proposed general concept leads to the formulation of the concept of an artificial atom and $H_2O$ model, an instance model necessary for conducting experiments in order to prove the possibility of spontaneous self-formation of complex programs from simpler ones.

We have also introduced graphic notation describing the formation of complex molecules from atoms by means of interconnections through "sinks" and "sources". This notation is a graphic representation of the "sinks-and-sources" algebra.

To test our theory, we have developed three types of interacting program components. They come in pure "sink" and "source" types or a hybrid "sink-and-source" type.

Due to the interactions during the experiments, these components spontaneously and unintentionally combined into more complex program structures. The resulting

structures can only pump data through themselves, but in the future their functionality may be developed.

The results of the experiments show that the main hypothesis of this article is correct and such spontaneous self-formation of program structures is actually possible under given conditions. The main purpose of the presented study has been achieved.

The results obtained are important for software engineering in order to increase the efficiency of the software development process and to make software flexible and capable of self-improvement during its life cycle. These results are also necessary for research in the fields of artificial life and artificial intelligence to study the self-formation of logical and program structures.

Many models of artificial chemistry are focused on the study and reproduction of reaction systems occurring in the real world and living systems. Examples of such models can be found in [1, 2, 6, 7, 16]. An example of software created on the basis of such a model of artificial chemistry and intended for chemical research is given in chapter 10 in [7].

However, our proposed model is one of the few models focused on the creation of new software engineering paradigms and technologies. The advantage of this model is that it allows compatibility with popular imperative programming tools and can be used for a wide range of applications.

But in general, the proposed models need more refinement. It is also necessary to continue the search for means to represent the repulsive force in a more obvious way. These questions provide material for further research.

## References


1. **Dittrich P., Ziegler J., Banzhaf W.** Artificial Chemistries — A Review, *Artificial Life*, 2001, vol. 7, no. 3, pp. 225—275.

2. **Banzhaf W., Yamamoto L.** *Artificial Chemistries,* Cambridge, Massachusetts; London, England, The MIT Press, 2015, 576 p.

3. **Kol'chugina E. A.** *Self-organization and self-development of programs in artificial chemistry models*, Kazan, OOO "Buk", 2019, 256 p. (in Russian).

4. **Eigen M., Schuster P.** *The Hypercycle: A Principle of Natural Self-Organization*, Berlin-Heidelberg-New York, Springer-Verlag, 1979, 98 p.

5. **Anchordoqui L. A., Chudnovsky E. M.** Can Self-Replicating Species Flourish in the Interior of a Star? *Letters In High Energy Physics,* 2020, vol. 2020, LHEP-166, pp. 1—4. DOI: 10.31526/LHEP.2020.166.

6. ***Artificial** Life: An Overview* / ed. by C. G. Langton, Cambridge, Massachusetts, The MIT Press, 1997, 336 p.

7. ***Artificial** Life Models in Software* / ed. by M. Komosinski, A. Adamatzky, London, Springer, 2009, 462 p.

8. **Kol'chugina E. A.** Spontaneous Emergence of Programs from "Primordial Soup" of Functions in Distributed Computer Systems, *Automatic Control and Computer Sciences*, 2018, vol. 52, no. 1, pp. 40—48. DOI: 10.3103/S0146411618010054.

9. **Von Neumann J.** *Theory of self-replication automata/* ed. by A. W. Burks, Urbana-London, University of Illinois Press, 1966, 416 p.

10. **Mitchell M.** Computations in Cellular Automata: A Selected Review, *Nonstandard Computation*, Weinheim, VCH Verlagsgesellschaft, 1998, pp. 95—140. DOI: 10.1002/3527602968.ch4.

11. **Fontana W.** Algorithmic Chemistry, *Artificial Life II, SFI Studies in the Sciences of Complexity, Vol. X* / ed. by C. G. Langton, C. Taylor, J. D. Farmer, S. Rasmussen, Redwood City, CA, Addison-Wesley, 1991, pp. 159—209.

12. **Kol'chugina E. A.** Model reproduction of non-equilibrium thermodynamics principles as a means to provide software self-development, *IOP Conference Series: Materials Science and Engineering; III International Scientific Conference: Modernization, Innovations, Progress: Advanced Technologies in Material Science, Mechanical and Automation Engineering (MIP-III 2021)*, 29$^{th}$—30$^{th}$ April 2021, Krasnoyarsk, Russian Federation, 2021, vol. 1155, p. 012054, DOI: 10.1088/1757-899X/1155/1/012054.

13. **Koshland D. E.** Application of a Theory of Enzyme Specificity to Protein Synthesis, *Proceedings of the National Academy of Sciences of the United States of America,* 1958, vol. 44, no. 2, pp. 98—104.

14. **Christiansen T., Torkington N.** *Perl Cookbook, Beijing-Cambridge-Koln-Paris-Sebastopol-Taipei-Tokyo*, O'Reilly & Associates, Inc., 1998. 794 p.

15. **Church A.** An Unsolvable Problem of Elementary Number Theory, *American Journal of Mathematics*, 1936, vol. 58, no. 2, pp. 345—363.

16. **Clark E. B., Hickinbotham S. J., Stepney S.** Semantic closure demonstrated by the evolution of a universal constructor architecture in an artificial chemistry, *Journal of the Royal Society Interface*, 2017, vol. 14, no. 130, article ID 20161033. DOI: 10.1098/rsif.2016.1033.

УДК 004, 004.3, 004.9, 519.6, 519.7, 519.8
DOI: 10.17587/prin.13.449-461

**А. Бернадотт,** канд. мед. наук, доц., bernadotte.alexandra@intsys.msu.ru,
Национальный исследовательский технологический университет "МИСиС",
ООО "Нейроспутник", Москва, Механико-математический факультет МГУ им. М. В. Ломоносова

# Структурная модификация конечного автомата для решения проблемы экспоненциального взрыва

*В ряде современных приложений, таких как системы обнаружения и предупреждения вторжений, экспертные знания формализуются в виде регулярных выражений, после чего проводится проверка принадлежности слова регулярному языку конечным автоматом. При этом задача понижения пространственной сложности при сохранении низкой временной сложности крайне актуальна. Представлен обзор современных решений данной задачи, основной идеей которых является переход от абстрактного конечного автомата, представленного таблично заданной функцией переходов, к структурному автомату, комбинирующему абстрактную часть, хранящуюся в памяти, и различные добавки типа битовых массивов и счетчиков.*

***Ключевые слова:*** *конечный автомат, экспоненциальный взрыв, сетевые системы обнаружения вторжения, структурная сложность, понижение сложности, регулярные языки, регулярные выражения*

## Введение

Принадлежность слова регулярному языку используется в широком спектре сетевых служб и служб безопасности на большинстве сетевых устройств, таких как маршрутизаторы, межсетевые экраны, коммутаторы уровня 7 и сетевые системы обнаружения вторжения (ССОВ).

Данные ССОВ зачастую интегрированы с базами регулярных выражений, составляющих регулярный язык, описывающий признаки атак. Детектирование вторжения проводится путем проверки трафика на наличие слов, принадлежащих данному регулярному языку. Большинство ССОВ с открытым исходным кодом, такие как Snort, используют сигнатурные базы Perl-совместимых регулярных выражений для реализации анализа сетевого трафика. Эффективность использования памяти в алгоритмах обработки сетевого трафика в сетевых устройствах является критической характеристикой в силу предпочтительного для скорости использования данными устройствами преимущественно статической памяти с произвольным доступом (SRAM).

На 2022 г. проверка принадлежности слова в трафике регулярному языку большинства ССОВ представляет сложность при использовании классического метода — распознавания языка детерминированным конечным автоматом (ДКА). Причиной данной сложности является увеличение числа состояний распознающего ДКА, что критично для памяти системы обнаружения вторжений.

В практическом использовании конечные автоматы в составе ССОВ могут комбинироваться с дополнительными аппаратными элементами, типа счетчиков, битовых массивов, переходов между медленной и быстрой памятью и другими элементами. В данной работе приведен обзор вариантов структурной модификации конечного автомата для решения проблемы увеличение числа состояний распознающего ДКА.

## Формальные понятия и определения

Основные определения даны в соответствии с работами [1, 2].

Язык $M$, $M \subseteq A^* \setminus \{\lambda\}$, называется регулярным, если его можно получить применением конечного числа операций объединения, произведения и итерации над регулярными языками вида $\varnothing$, $\{a\}$, $a \in$ A. Формально, регулярный язык над алфавитом $A$ задается рекурсивно:

1) $\varnothing$, $\{\lambda\}$ и $\{a\}$ — регулярные языки, где $\lambda$ — пустое слово, $a \in A$;

2) если $M_1$ и $M_2$ — регулярные языки, то объединения $M_1 \cup M_2 = \{\alpha_1 \cup \alpha_2 \mid \alpha_1 \in M_1, \alpha_2 \in M_2\}$ и конкатенация $M_1 \cdot M_2 = \{\alpha_1\alpha_2 \mid \alpha_1 \in M_1, \alpha_2 \in M_2\}$ языков $M_1$ и $M_2$, а также итерация или звезда Клини $M_1^* = \{\lambda\} \cup \{\alpha_1 \cdot \ldots \cdot \alpha_n \mid \alpha_i \in M_1,\ i \in \mathbb{N}\}$ — регулярные

языки, где регулярность языка устанавливается за конечное число операций объединения, конкатенации и итерации [1, 2].

Формулу, которая описывает последовательность операций, принято называть "регулярным выражением", задающим данный регулярный язык. Формально, регулярные выражения представляют собой слова над алфавитом $A \cup \{\cup, (,), \cdot, *\}$, определяемые следующим образом:

1) $\varnothing$, $\lambda$, $a$ — регулярные выражения, где $a \in A$;

2) если $R_1$, $R_2$ — регулярные выражения, то слова $(R_1 \cup R_2)$, $(R_1 \cdot R_2)$, $(R_1)^*$ — регулярные выражения.

Регулярный язык, задаваемый регулярным выражением, определяется естественным образом, путем выполнения последовательности операций объединения, конкатенации и итерации, задаваемой данным регулярным выражением.

На практике в базах Snort, Zeek, Cisco регулярные выражения представляют с использованием нотации PCRE (*Perl Compatible Regular Expressions*) [3]. Приведем PCRE-обозначения, которые будем использовать в работе:

"." — произвольный символ из алфавита;

".*" — ноль или более вхождений произвольного символа из алфавита;

"$a$+" — одно или более вхождений "$a$";

"{$n$}" — число вхождений, равное $n$;

"{$n$, $m$}" — число вхождений от $n$ до $m$;

"|" — логическое "объединение";

"^", "¬" — логическое "не";

"[]" — групповой символ;

"[0−9]" — любой цифровой символ;

"[$A$ − $Za$ − $z$0 − 9]" — любой буквенный или цифровой символ.

Формально, ДКА $V$ называется набор $V = (A, Q, Q_B, \varphi)$, где $A \neq \varnothing$ — входной алфавит; $Q \neq \varnothing$ — конечное множество состояний; $Q_B \subseteq Q$ — множество заключительных состояний; $\varphi$ — функция переходов; $\varphi$: $A \times Q \to Q$.

Инициальный конечный автомат имеет отдельно выделенное начальное состояние, в котором начинается работа автомата: $V = (A, Q, Q_B, \varphi)$, где $q_0$ — начальное состояние автомата; $q_0 \in Q$.

Недетерминированный конечный автомат (НДКА) позволяет осуществлять переход из одного и того же состояния под действием одной и той же буквы в разные состояния и представляет собой набор $V = (A, Q, Q_B, \varphi, q_0)$, где $A \neq \varnothing$ — входной алфавит; $Q \neq \varnothing$ — конечное множество состояний; $Q_B \subseteq Q$ — множество заключительных состояний; $\varphi$ — функция переходов, $\varphi$: $A \times Q \to 2^Q \setminus \{\varnothing\}$; $q_0 \in Q$ — начальное состояние автомата.

Согласно теореме об эквивалентности детерминированных и недетерминированных конечных автоматов, всякий язык принимается некоторым НДКА тогда и только тогда, когда этот язык принимается некоторым ДКА [4, 5]. Кроме того, существует алгоритм построения из НДКА эквивалентного ему ДКА [4, 5].

Конечные автоматы рассматриваются как устройства, распознающие (и принимающие) некоторые множества входных слов над конечным алфавитом. Подмножество $M$, $M \subseteq A^* \setminus \{\lambda\}$, где $\lambda$ — пустое слово, называют языком над алфавитом $A$ [1, 2]. Пусть $V_q = (A, Q, Q_B, \varphi, q_0)$ — инициальный автомат, $Q_B \subseteq Q$. Множество $M$ (язык $M$) = $\{\alpha: \alpha \in A^*, \varphi(q, \alpha) \in Q_B\}$ называется представимым в автомате $V_q$ с помощью подмножества заключительных состояний $Q_B$, или говорим, что автомат $V_q$ принимает $M$ посредством $Q_B$. Множество $M$ (язык $M$) называется представимым.

В российской литературе представление языка проводится не посредством состояний, а посредством выходов [1, 2]. Пусть $V_q = (A, Q, B, \varphi, \psi, q_0)$ — инициальный автомат, где $A \neq \varnothing$ — входной алфавит; $Q \neq \varnothing$ — конечное множество состояний; $B$ — выходной алфавит; $\varphi$ — функция переходов, $\varphi$: $A \times Q \to Q$; $\psi$ — функция выходов; $\psi$: $A \times Q \to B$. Множество $M$ (язык $M$) = $\{\alpha: \alpha \in A^*, \psi(q, \alpha) \in B'\ B' \subseteq B\}$ называется представимым в автомате $V_q$ с помощью подмножества $B'$ входных символов, или говорим, что автомат $V_q$ принимает $M$ посредством $B'$. Распознавание посредством выходов или посредством распознавания состояниями является эквивалентным [1, 2].

В прикладной задаче проверки принадлежности слова регулярному языку значимым является пространственная сложность реализующего проверку конечного автомата (КА) — число состояний распознающего КА (и объем необходимой для проверки памяти), и временная сложность распознавания — число переходов из одного состояния в другое (соответствующих тактам работы КА). Принято считать, что вычисление значения функции переходов автомата имеет константную сложность — сводится к извлечению значения из памяти.

Детерминированный конечной автомат имеет оптимальную по порядку временную сложность, однако число состояний автомата (пространственная сложность) может расти экспоненциально от длины регулярного выражения.

Недетерминированный конечный автомат обладает низкой пространственной сложностью, линейно зависимой от длины регулярного выражения. При этом временная сложность является основным недостатком НДКА, так как время обработки одного символа входного слова, вообще говоря, также линейно зависит от длины регулярного выражения.

При переходе от недетерминированного автомата к детерминированному временная сложность обработки входного символа становится константной. Однако в общем случае при входном алфавите мощности более или равной двум мощность множества состояний экспоненциально возрастает от длины регулярных выражений [1, 2, 6].

Связь между регулярным языком и автоматом описывается теоремой Клини: слово лежит в языке тогда и только тогда, когда заключительное состояние соответствующего автомата лежит в выделенном подмножестве.

## Проблема экспоненциально растущего числа состояний конечного автомата

В прикладных задачах распознавания регулярного языка значительной проблемой является проблема экспоненциально растущего числа состояний распознающего ДКА от длины регулярных выражений распознаваемого языка [1, 6].

К признакам, вызывающим экспоненциальный взрыв, относятся: использование в регулярных выражениях сочетаний ".+" и ".*" и "считающих" выражений типа ".{*n*}" и "[$\neg a_i$]{*n*}".

Отдельное внимание уделяется проблеме экспоненциального взрыва для языка $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где $\alpha_i$, $\beta_i$ — непустые слова в алфавите $A$. Известно, что в общем случае для представления языка из данного класса требуется экспоненциальное по $n$ число состояний ДКА [7].

Проблемой экспоненциального взрыва исследователи занимаются более 20 лет. Существуют следующие три основных подхода к решению этой проблемы с использованием КА.

- Ограничение на сигнатуры, задаваемые экспертами. Данный метод предполагает экспертный подбор регулярных выражений, позволяющий сократить число состояний данного КА. Минусом данного подхода являются низкая автоматизация подхода и его высокая трудоемкость. При этом возможно как расширение, так и сужение распознаваемого языка после экспертной работы.
- Модификация распознаваемого регулярного языка. Данный подход предполагает появление в решении практической задачи распознавания ошибок первого и второго рода. При этом метод дает значимое с практической точки зрения уменьшение сложности КА за счет расширения регулярных языков [8].
- Модификация структуры КА. Данный подход заключается в модификации структуры КА без изменения распознаваемого языка и может быть реализован:

— как модификация КА через применение алгоритмов сжатия [9];

— как модификация структуры КА через добавление специальных структурных элементов.

В настоящей работе предметом интереса является модификация структуры КА через добавление специальных структурных элементов, которая не требует изменения распознаваемого языка.

## Существующие структурные расширения конечных автоматов

Самыми распространенным структурными элементами, расширяющими возможности КА, являются счетчики и дополнительные элементы памяти, хранящие определенные свойства КА. В данном разделе приведен список существующих решений, основной идеей которых является переход от абстрактного КА, представленного таблично заданной функцией переходов, к структурному автомату, комбинирующему абстрактную часть, хранящуюся в памяти, и различные добавки типа битовых массивов, счетчиков, подсхем. При этом пространственная сложность определяется как сумма объема памяти, занимаемой абстрактной частью, и сумма сложностей добавок (т. е. схемная сложность). Экспоненциальное число состояний перемещается в структурную часть, так что общая сложность остается приемлемой.

Введем термин "метасостояние конечного автомата". Метасостояние конечного автомата — дополнительный параметр КА в наборе, описывающий дополнительные структурные элементы КА.

Детерминированный абстрактным конечным автоматом с метасостоянием называется набор $V = (A, Q, Q_B, M, \varphi, \omega)$, где $A$, $Q$, $Q_B$, $M$ — конечные множества: входной алфавит, алфавит состояний, подмножества заключительных состояний и алфавит метасостояний соответственно; $\varphi$ — функция переходов, $\varphi: A \times Q \times M \to Q$; $\omega$ — функция смены метасостояния, $\omega: A \times Q \times M \to M$.

Хранение значения дополнительного параметра КА, отвечающего за выделение метасостояний, чаще всего осуществляется посредством отдельных битов, битовых массивов, битовых масок и счетчиков. Экспоненциальное число состояний перемещается в структурную часть, так что общая сложность остается приемлемой. Наглядно такое структурное перестроение будет изложено ниже.

Рассмотрим существующие принципиальные решения проблемы экспоненциального взрыва с использованием дополнительных структурных элементов.

### *Двойной конечный автомат с использованием быстрой и медленной памяти*

Занимаясь проблемой экспоненциального взрыва в приложении к распознаванию вредоносного трафика, Кумар и соавторы [10] вводят несколько понятий, характеризующих недостатки использования классического ДКА в контексте проблемы экспоненциального взрыва. Так, "бессонницей" (в оригинале *insomnia*) авторы называют расточительное хранение всех состояний автомата, вне зависимости от вероятности их активации [10].

Действительно, при анализе трафика КА большинство состояний активируются редко, и вероятность активации состояния падает по мере удаления от начального состояния КА.

Авторами предлагается делить регулярное выражение на префиксную и суффиксную части, которые определяются вероятностью перехода КА в состояния, соответствующие суффиксной части. К высоковероятным состояниям относят состояния, описывающиеся префиксами регулярных выражений, к маловероятным состояниям относятся те, которые описываются хвостовыми частями (суффиксами) регулярных выражений. Авторами предлагается хранить переходы в высоковероятные состояния ДКА в быстрой памяти, а переходы в маловероятные состояния — "в состоянии сна" (в медленной памяти). Переключение из быстрой памяти в медленную должно осуществляться при необходимости — при срабатывании триггера. При этом триггером является распознавание префикса подслова быстрой частью ДКА.

Структурным элементом решения является медленная память с переключателями между быстрой и медленной памятью. Особенность такого хранения состояний дает выгоду по объему затрачиваемой быстрой памяти, при этом обращение к медленной памяти замедляет работу данной конструкции при сравнении с классическим ДКА. Однако предварительный статистический анализ трафика позволяет снизить необходимость обращения к медленной памяти. Таким образом, конструкция позволяет снизить объем затрачиваемой быстрой памяти без существенных временных потерь скорости работы.

На рис. 1 приведен пример двойного КА из работы Кумар и соавторов, принимающего язык $L$, $L = R_1 \cup R_2 \cup R_3$, представленный объединением следующих регулярных выражений:

$R_1 = .*[gh]d[\neg g]*ge$, $R_2 = .*fag[\neg i]*i[\neg j]*j$, $R_3 = .*a[gh]i[\neg l]*[ae]c$.

Модификация классического ДКА авторами предложена для прикладной задачи обнаружения вторжений при анализе трафика, и сама модификация ориентируется на предварительный анализ нормального и вредоносного трафика. Вопрос о разделении состояний на вероятные и маловероятные предлагается авторами решать эмпирически, а именно через рассмотрение распределения вероятностей различных входных символов для конкретного трафика, анализируемого устройством с описываемым автоматом [10—12].

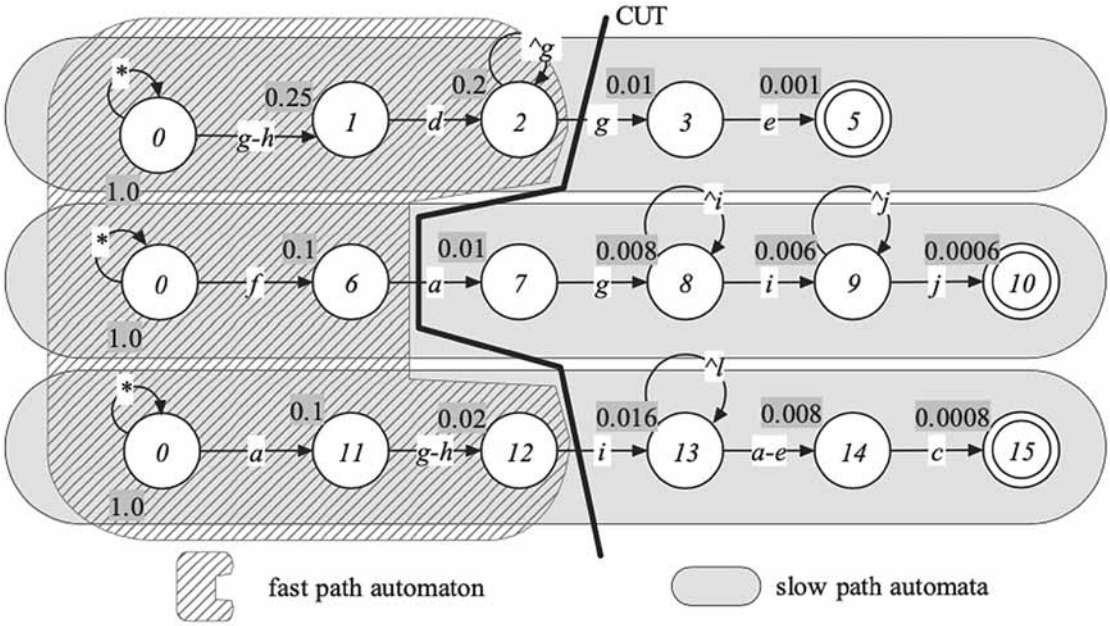


**Рис. 1. Диаграмма двойного КА с изображением быстрой (слева) и медленной частей (справа) памяти. Диаграмма соответствует принимаемому языку $L$, $L = R_1 \cup R_2 \cup R_3$, $R_1 = .*[gh]d[\neg g]*ge$, $R_2 = .*fag[\neg i]*i[\neg j]*j$, $R_3 = .*a[gh]i[\neg l]*[ae]c$. Над стрелками перехода отмечена вероятность перехода, полученная при предварительном анализе трафика, двойной линией обозначены принимающие состояния [10]**

Следует отметить, что в системе обнаружения вторжений, в состав которой входит предложенный двоичной КА, требуется предварительно вычислять вероятность совпадения префиксов разной длины при нормальном и аномальном трафике и на основе полученных данных хранить состояния в медленной или быстрой памяти. Таким образом, чтобы использовать двойной КА в прикладной задаче недостаточно использовать существующую базу регулярных выражений, маркирующих вредоносный трафик. Требуется "дообучить" систему на собранном датасете, где трафик будет размечен относительно частоты активации состояний соответствующего КА. Данное свойство несколько затрудняет использование интересного решения коллег.

### *Детерминированный конечный автомат с битовым массивом*

Существует несколько решений, использующих ДКА в комбинации с хранением дополнительной информации в памяти в виде битового массива.

**Детерминированный конечный автомат с памятью (битовым массивом).** Неспособность КА запоминать "посещенные состояния" Кумар и соавторы называют "амнезией" (в оригинале *amnesia*) и предлагают хранить метасостояние автомата в виде нескольких флагов (битов) в дополнение к текущему состоянию автомата [10]. Автомат с такой модификацией авторами назван "ДКА с памятью" (в оригинале *History based Finite Automaton* или сокращенно — H-FA) [10].

Формально, ДКА с памятью представлен набором, H-FA = ($A$, $Q$, $q_0$, $Q_B$, φ, $H$), где $A$ — входной алфавит; $Q$ — алфавит состояний; $q_0$ — начальное состояние; $Q_B$ — множество принимающих состояний; $H$ — битовый массив; φ — функция перехода, φ: $A \times Q \times H \rightarrow Q \times H$ [10]. Битовый массив принимает соответствующее состоянию КА значение, равное 1, если в данное состояние ранее был осуществлен переход, и значение, равное 0, если такого перехода не было.

Детерминированный конечный автомат с памятью структурно расширяет ДКА с помощью вспомогательной памяти (битового массива), осуществляя работу сложных переходов в зависимости от хранимой в памяти информации. Выигрыш от использования битового массива заключается в уменьшении числа необходимых состояний и числа переходов за счет хранения дополнительной информации о "посещенных" состояниях. На рис. 2, *б* приведен пример ДКА с памятью (H-FA) из работы Кумар и соавторов, принимающий язык $L = R_1 \cup R_2$, $R_1 = .*ab[\neg a]*c$, $R_2 = .*def$, и дано сравнение с классическим ДКА (рис 2, *а*), демонстрирующее пространственную выгоду использования битового массива.

Модификация ДКА с памятью под названием HASIC, предложенная группой авторов [13], делает работу системы обнаружения вторжений в несколько раз более эффективной.

**Расширенный детерминированный конечный автомат с битовым массивом.** Другая группа исследователей [14] также использовала идею с битовым массивом для хранения метасостояния ДКА. Группа сфокусировала свое внимание на определенном классе языков, вызывающем экспоненциальный взрыв и накладывающем значительное ограничение на эффективность и скорость работы систем обнаружения вторжений — $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где $\alpha_i$, $\beta_i$ — слова над алфавитом $A$ [14].

Представленный "расширенный ДКА" (в первоисточнике — *extended finite automata* (XFA)) решает проблему экспоненциального взрыва для языков класса: $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где слова $\alpha_i$, $\beta_i$ имеют определенные ограничения [14]. При этом расширенный ДКА позволяет снизить число состояний распознающего ДКА с $O(nl2^n)$ числа состояний нативного ДКА до $O(nl)$ состояний расширенного ДКА, где $l$ — максимальная длина слов $\alpha_i$, $\beta_i$. Формально расширенный ДКА — это ($A$, $Q$, $D$, φ, $U_\varphi$, ($q_0$, $d_0$), $F$), где $A$ — входной алфавит; $Q$ — алфавит состояний; φ: $Q \times A \rightarrow Q$ — функция перехода; $D$ — битовый массив с начальными нулевыми значениями; $U_\varphi$: $Q \times A \times D \rightarrow D$ — функция обновления битового массива; ($q_0$, $d_0$) — начальные состояния из $Q$ и $D$; $F \subseteq Q \times D$ — множество принимающих конфигураций.

Дополнительным структурным элементом данного решения является битовый массив, который фиксирует метасостояния ДКА в отношении распознанного подслова $\alpha_i$. Снижение пространственной сложности реализуется через сохранение дополнительного бита в специальном структурном элементе — битовом массиве, при распознавании автоматом первого слагаемого в классе языков $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где слова $\alpha_i$, $\beta_i$ не являются подсловами друг друга [14].

Данная модификация расширенного ДКА позволяет "запомнить", что первое слагаемое $\alpha_i$ из $.*\alpha_i.*\beta_i.*$ было распознано, и перевести ДКА в соответствующее метасостояние. При нахождении расширенного ДКА в определенном метасостоянии и при распознавании второго слагаемого — $\beta_i$, происходит распознавание всего слова —$.*\alpha_i.*\beta_i.*$. При этом число хранимых бит в битовом массиве соответствует числу первых сомножителей.

Смит и коллеги применяют расширенный ДКА к еще одному классу языков. На рис. 3 показан

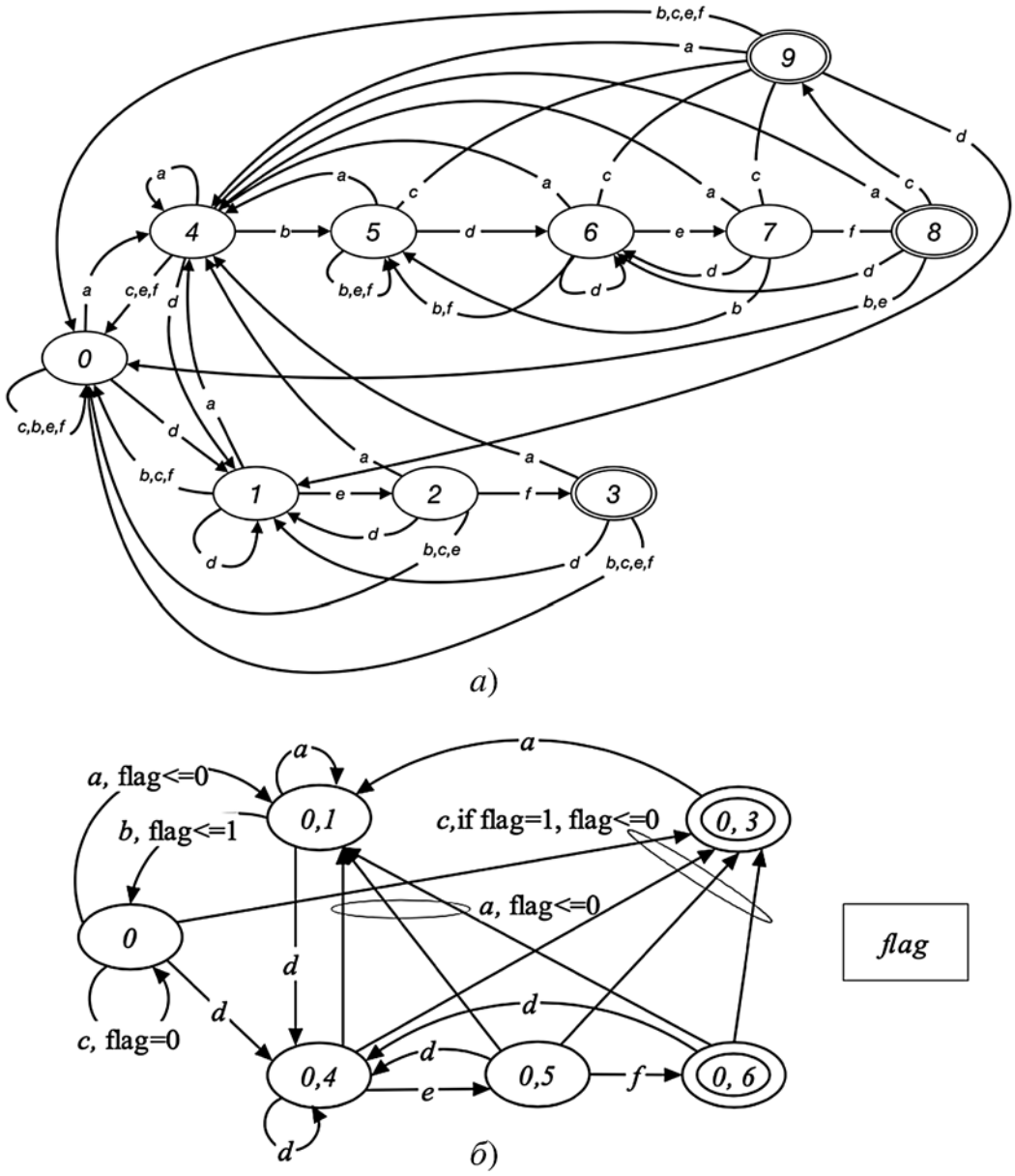


**Рис. 2. Детерминированный конечный автомат, распознающий язык $L = R_1 \cup R_2$, $R_1$ =.*ab[¬a]*c, $R_2$ =.*def над алфавитом *A* (*а*) и ДКА с памятью, распознающий язык $L = R_1 \cup R_2$, $R_1$ =.*ab[¬a]*c, $R_2$ =.*def над алфавитом *A* (*б*). Память реализована хранимым битовым массивом. На рисунке бит, названный *flag*, принимает значение 1 при переходе из состояния 0,1 в состояние 0 по символу *b*, в обратном случае *flag* принимает значение 0 [10]**

переход от обычного ДКА, распознающего язык $L = \{(.* ab.* cd) \cup (.* ef.* gh)\}$ над алфавитом {*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*} к расширенному ДКА [14].

Такое решение требует существенных ограничений, накладываемых на распознаваемый язык. Авторы утверждают, что необходимо, чтобы слова $\alpha_i$ и $\beta_i$ не являлись подсловами друг друга. Требует детального рассмотрения возможность ослабления данного ограничения. Так, если сначала проводить проверку на активный бит, а потом присваивать его (рис. 4), то подобное ограничение не требуется.

При этом у предложенного решения останется классическая проблема алгоритмов борьбы с экспоненциальным взрывом при рассмотрении данного класса языков $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$: если суффикс $\alpha_i$ является префиксом $\beta_i$, то данная представленная конструкция расширенного ДКА с битовым массивом расширяет принимаемый язык. Дадим название отмеченной проблеме: "проблема расширения языка". Примеры, иллюстрирующие данную проблему расширения языка, представлены далее.

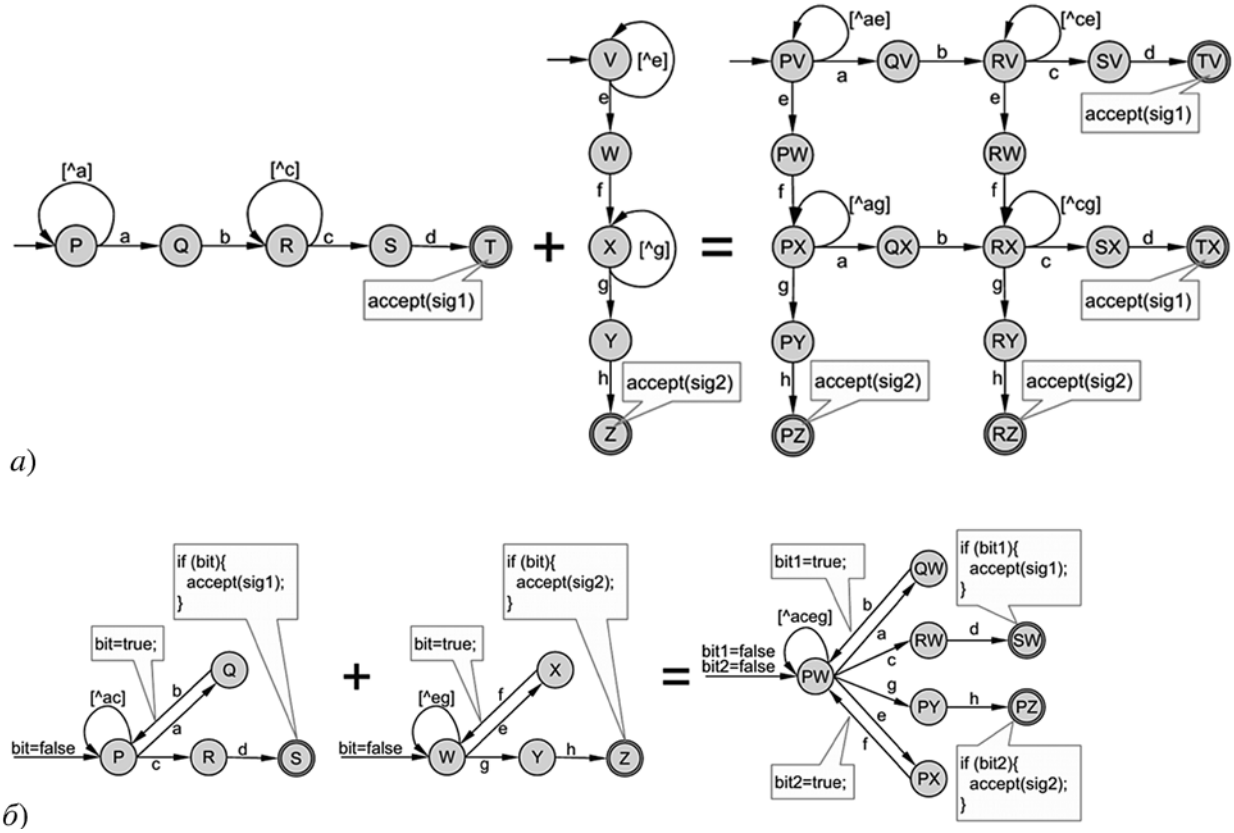


**Рис. 3. ДКА (*а*) и расширенный ДКА (*б*), распознающие язык $L$ = {(.**ab*.**cd*) ∪ (.**ef*.**gh*)} над алфавитом {*a, b, c, d, e, f, g, h*} [14]**

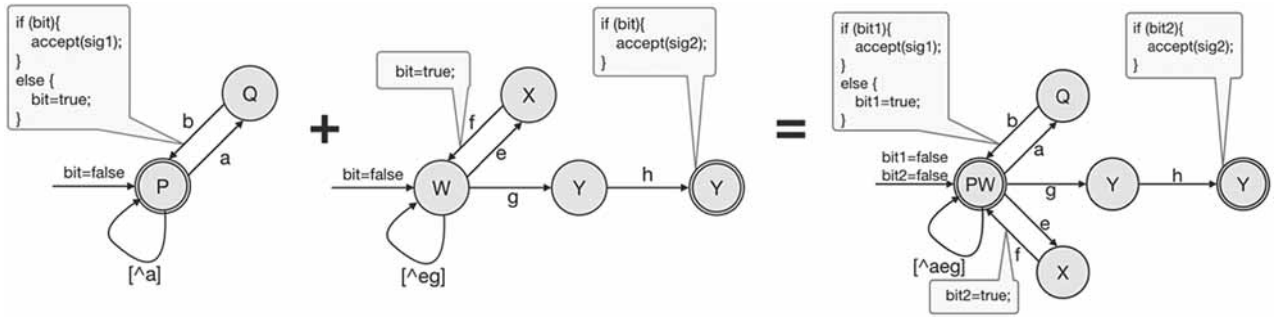


**Рис. 4. Расширенный ДКА, распознающий язык $L$ = {(.* *ab*.* *ab*) ∪ (.* *ef*.* *gh*)} над алфавитом {*a, b, c, d, e, f, g, h*}**

### *Гибридные конечные автоматы*

Сочетание ДКА и НДКА дает ощутимое преимущество для решения проблемы экспоненциального взрыва, позволяя комбинировать положительные с точки зрения экономии ресурсов и времени свойства детерминированности и недетерминированности. Существует несколько предложений использования комбинации ДКА и НДКА в контексте проблемы экспоненциального взрыва [15, 16].

**Детерминированный конечный автомат с линейным автоматом.** Лиу и Ву отмечают, что причина проблемы экспоненциального взрыва ДКА заключается в большом числе "независимых состояний" соответствующего НДКА (назовем "нативный НДКА", НДКА = ($A$, $Q$, $Q_B$, φ, $q_0$), из которого строится ДКА, где два состояния $v_i$ и $v_j$ нативного НДКА являются независимыми, если существуют три множества активных состояний α, β и γ, такие что $v_i \in \alpha$, $v_j \notin \alpha$, $v_i \notin \beta$, $v_j \in \beta$, $v_i$, $v_j \in \gamma$. Для решения проблемы экспоненциального взрыва авторами представлена конструкция, названная "двойной конечный автомат" [15]. Чтобы не путать с описанным ранее решением, назовем данную конструкцию "детерминированный конечный автомат с линейным автоматом".

Детерминированный конечный автомат с линейным автоматом состоит из ДКА и дополнительного структурного элемента — конечного линейного автомата. Классический ДКА данной конструкции расширяется по средствам "расширенных" и "условных" переходов, которыми данный ДКА взаимодействует с линейным. Такой ДКА авторы называют "расширенным ДКА". Линейный конечный автомат (ЛКА) — это НДКА, представляющий собой КА с переходами такими, что из каждого состояния возможен переход в себя и в единственное другое состояние, при этом в каждое состояние возможен переход из единственного состояния, отличного от данного. "Линейность" автомата позволяет упорядочить состояния и хранить переходы в виде битового массива (битовой маски), объемом $2|A||Q|$ бит, где $A$ — входной алфавит; $Q$ — алфавит состояний НДКА. Состояния упорядочены таким образом, что если существует переход из состояния $u$ (представлено $i$-м битом в битовом массиве) в состояние $v$, то состояние $v$ должно быть представлено $(i-1)$-м битом, находящимся в битовом массиве справа от $i$-го бита. Для каждого символа $a_i$ из алфавита $A$ формируются два битовых массива длиной равной $|Q|$: определяющие переход без смены состояния (битовая маска $self[a_i]$, где $self[a_i][j] = 1 \Leftrightarrow q_j \in \varphi(q_j, a_i)$), и переход в другое состояние по данному символу (битовая маска $next[a_i]$, где $\text{next}[a_i][j] = 1 \Leftrightarrow q_j - 1 \in \varphi(q_j, a_i)$).

Для иллюстрации на рис. 5 приведем решение авторов для языка $L = (.+B.^*) \cup (.^*[A-Q] + [I-Z])$. Решение авторов приводит к изменению пространственной сложности, определяемой числом состояний КА.

В один момент времени может существовать один активный битовый массив $L$, представляющий собой массив длиной $|Q|$ и фиксирующий метасостояние автомата, выражающееся в наборе активных состояний. Активный битовый массив $L$ можно вычислить с помощью следующих побитовых операций при получении на вход символа $a_i$ из алфавита $A$: $(L \cap self[a_i]) \cup [(L \cap next[a_i]) >> 1]$, где $\cap$, $\cup$ и $>>$ побитовые операторы "и", "или", и "сдвиг вправо" соответственно. Активный битовый массив является меткой расширенного перехода, объединяющего ДКА и ЛКА. Условный переход осуществляется при нахождении автомата в определенном метасостоянии, определяемом текущим активным битовым массивом. Каждый переход расширенного ДКА должен быть закодирован двумя битовыми массивами — кодирующими индекс следующего состояния расширенного ДКА и состояния ЛКА.

Для каждого символа из алфавита $A$, поступающего на вход, двойной КА выполняет следующие действия: 1) в соответствии с поступающим символом из алфавита $A$ ЛКА и расширенный ДКА совершают переходы, при этом ЛКА определяет новое метасостояние двойного автомата, закодированное

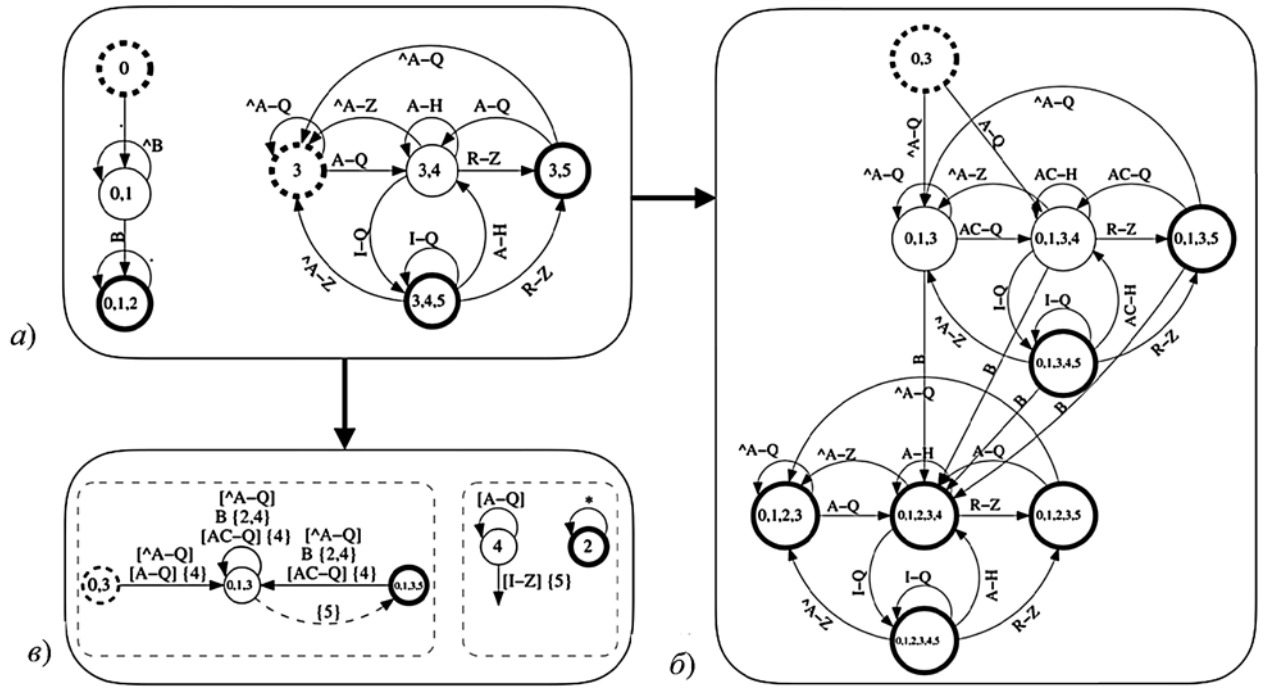


**Рис. 5. Два ДКА, распознающих языки $L_1 = . + B.^*$ и $L_2 = .^*[A-Q] + [I-Z]$, требующих три и четыре состояния соответственно (*а*); ДКА, распознающий язык $L = (. + B.^*) \cup (.^*[A-Q] + [I-Z])$, представленный девятью состояниями (*б*); двойной КА с линейным автоматом, распознающий язык $L = (.+B.^*) \cup (.^*[A-Q] + [I-Z])$, представленный пятью состояниями (*в*) [15]**

как "активный битовый массив *L*"; 2) дополнительные состояния ЛКА могут быть установлены как активные, если они указаны переходом, выполняемым расширенным ДКА; 3) расширенный ДКА совершает условный переход, если на это указывает переход, выполняемый ЛКА.

Метасостояние КА определяется активностью дополнительного линейного автомата, закодированного как активный битовый массив *L*. Таким образом, реализация расширенного ДКА с ЛКА позволяет использовать описанную ранее схему фиксации метасостояния через хранение битов.

Говоря о структурной организации конструкции, хранение битовых массивов может осуществляться через хранение в кеш-памяти (или встроенной памяти), обеспечивающей быстрый доступ. Однако само построение оптимального двойного КА с точки зрения минимизации числа состояний является задачей, не имеющей оптимальных алгоритмов.

### *Конечные автоматы со счетчиками*

При реализации как НДКА, так и ДКА существует проблема неэффективного подсчета вхождения определенных подвыражений. Всякий раз, когда регулярное выражение содержит ограничение длины *k* для подвыражения типа *R*{*k*}, число состояний, требуемых подвыражением *R*, умножается на *k*. Большинство авторов предлагают решать данную проблему неэффективного использования КА через добавление счетчиков. Однако в данном обзоре решений приводится использование счетчиков и для решения проблемы экспоненциального взрыва.

**Детерминированный конечный автомат с битовым массивом и счетчиками.** Кумар и соавторы выделяют недостаток КА, связанный с появлением в принимаемом регулярном языке *R*{*k*} и *R*{*n*, *k*}, где *R* — регулярное выражение, специальным названием — "акалькулия" (в первоисточнике *acalculia*), в силу которой как НДКА, так и ДКА не могут эффективно подсчитывать вхождения определенных подвыражений [10].

Для решения проблемы экспоненциального взрыва авторами [10] предложен ДКА с блоком дополнительной памяти и блоком счетчиков (в первоисточнике — *History based counting finite Automata* или H-cFA), хранящих метасостояния автомата в виде набора битов. Кумар и соавторы дополняют конструкцию ДКА битовым массивом вместо тех состояний, которые вызывают экспоненциальный взрыв типа ".*" и ".+". Счетчики дополняют те состояния, которые соответствуют подвыражениям типа *R*{*k*} и *R*{*n, k*}, где *R* — регулярное выражение [10].

Формально, ДКА с памятью и счетчиками представлен набором H-cFA, H-cFA = (*A*, *Q*, *H*, *C*, $Q_B$, φ, $q_0$), где *A* — входной алфавит; *Q* — алфавит состояний; *H* — множество конфигураций, фиксирующих метасостояние автомата через дополнительный бит; *C* — множество конфигураций, фиксирующих метасостояние автомата через счетчики; φ — функция перехода, φ: $A \times Q \times H \times C \rightarrow Q \times H \times C$; $Q_B$ — множество принимающих состояний [10]. Пространственный выигрыш осуществляется за счет уменьшения мощности алфавита состояний (и требуемого объема табличной организации переходов) и оптимизации хранения переходов в виде счетчиков.

На рис. 6 приведен пример ДКА с памятью и счетчиками (H-cFA) из работы Кумар и соавторов, принимающий язык $L = R_1 \cup R_2$, $R_1$ =.*ab*[¬*a*]{4}*c*, $R_2$ =.**def*.

Практическое применение авторами [10] ДКА с памятью и счетчиками на существующих датасетах систем обнаружения вторжений привело к уменьшению числа состояний на порядок по сравнению с классическим ДКА.

Следует обратить внимание на особенность представленных Кумаром и соавторами структурно-модифицированных автоматов: данные автоматы имеют ограничение на правильность распознавания языков, что ранее было названо "проблемой расширения языка". Данная проблема расширения языка приводит к ложным срабатываниям в системе обнаружения вторжений.

Авторы интересного решения ДКА с памятью и счетчиками не акцентировали внимание

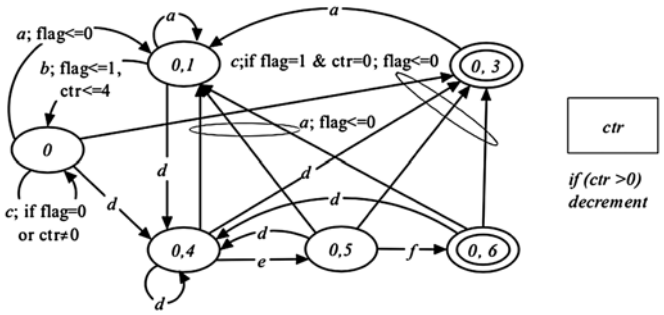


**Рис. 6. Детерминированный конечный автомат с памятью и счетчиками (H-cFA), распознающий язык (.**ab*[¬*a*]{4}*c*) ∪ (. * *def*) над алфавитом A. Счетчик на рисунке обозначен как *ctr*.**
**Память реализована хранимым битовым массивом. На рисунке бит, названный *flag*, принимает значение 1 при переходе из состояния 0,1 в состояние 0 по символу *b*, в обратном случае *flag* принимает значение 0. Счетчик *ctr* принимает значение 4 тогда и только тогда, когда реализуется переход из состояния 0,1 в состояние 0 по символу *b*. Далее каждый такт счетчик уменьшает свое значение на единицу. Сброс состояния счетчика до нулевого происходит при переходе из состояния 0 в состояние 0,1 по символу *a*. Условный переход реализует переход из состояния 0 в 0,3 тогда и только тогда, когда бит *flag* принимает значение 1 и счетчик *ctr* принимает значение 0 [10]**

на данной проблеме расширения языка. Однако незначительная модификация регулярного языка, который распознается ДКА с памятью (H-FA) на рис. 2, приводит тому, что предложенное решение дает ложные срабатывания в прикладной задаче обнаружения вторжений.

Так, преобразование языка $L = R_1 \cup R_2$, $R_1 = .*ab[\neg a]\{4\}c$, $R_2 = .*def$ посредством изменения первого регулярного выражения из $R_1 = .*ab[\neg a]\{4\}c$ в регулярное выражение $R' = .*ab[\neg a]\{4\}bc$, дает язык $L' = R'_1 \cup R_2$, где $R_2 = .*def$, при этом модифицированный ДКА с памятью (H-FA) будет принимать не $L'$, а язык $L \cup L'$, что и будет обеспечивать ложные срабатывания системы обнаружения вторжений.

**Детерминированный конечный автомат со счетчиками для решения проблемы экспоненциального взрыва и проблемы расширения языка**. Продолжением идеи расширенного ДКА явилась конструкция авторов Бернадотт и Галатенко [17]. Предложение авторов ориентировано на решение проблемы экспоненциального взрыва для языков следующего класса: $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$, где слова $\alpha_i$, $\beta_i$ — непустые слова в алфавите $A$. При этом конструкция решает проблему расширения языка.

Данная конструкция позволяет избежать экспоненциального взрыва пространственной сложности через разбиение КА на две компоненты: "абстрактную", функционирование которой задается таблично, а сложность определяется как объем памяти, требующейся для хранения таблицы значений, и "структурную", сложность которой определяется как число элементов в схеме [17].

Общая схема конструкции представлена на рис. 7. Автомат состоит из трех блоков. Блок 1 с помощью выходов распознает язык $L_1 \cup L_2$, где $L_1 = \bigcup_{i=1}^{n}(.*\alpha_i)$, $L_2 = \bigcup_{i=1}^{n}(.*\beta_i)$. Выходной алфавит данного блока — множество двоичных век-

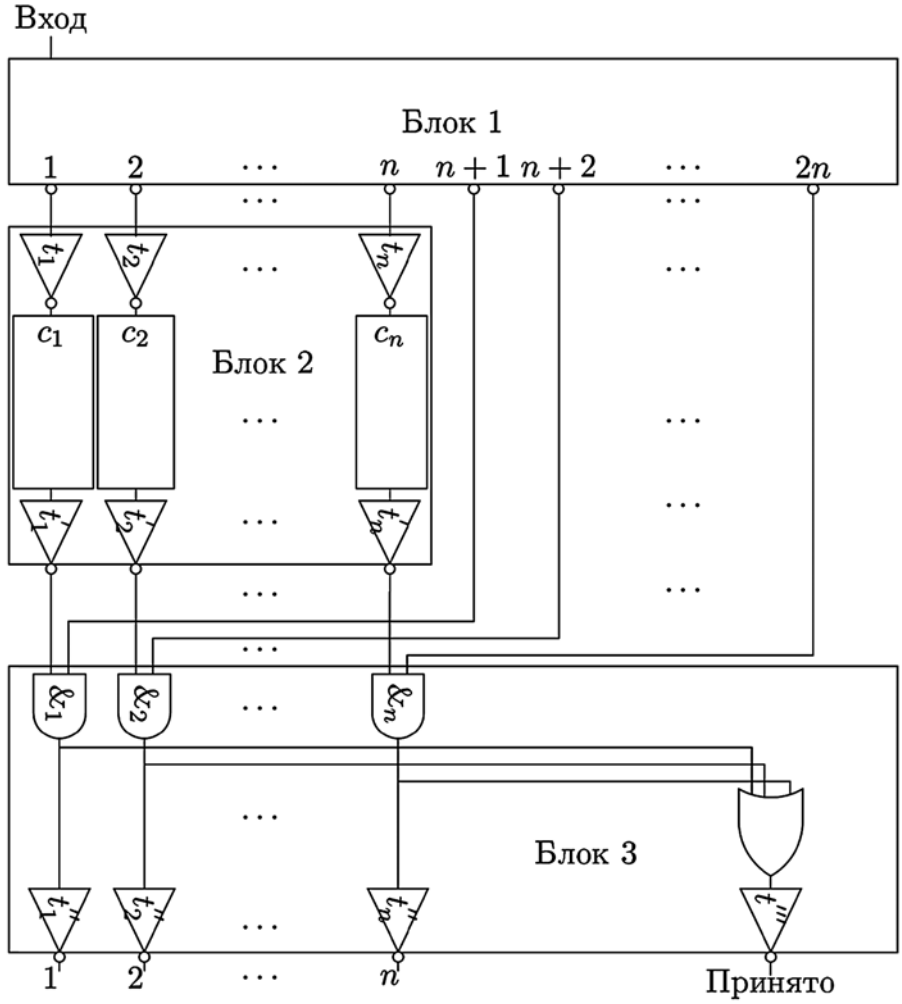


**Рис. 7. Аппаратная конструкция, $t_i$, $t'_i$, $t''_i$ $t'''_i$ — триггеры; прямоугольники $c_1$, ..., $c_n$ блока 2 — счетчики, отсчитывающие длину слов, соответствующих $\beta_i$ языка $L_2 = \cup_{i=1}^{n} = 1.*\beta_i$ [17]**

торов длины $2n$. Для $1 \leqslant i \leqslant n$, $i$-я компонента выхода обращается в 1 тогда и только тогда, когда входное слово принадлежит языку $.*\alpha_i$; если же $n+1 \leqslant i \leqslant 2n$, то равенство $i$-й компоненты выхода 1 эквивалентно принадлежности входного слова языку $.*\beta_i$. Первые $n$ компонент выхода блока 1 подаются на вход блока 2 и включают счетчики, отсчитывающие длины соответствующих слов $\beta_i$. Если требуемая длина достигнута, единичный сигнал передается на вход блока 3, вычисляющего конъюнкции выходов блока 1 с номерами от $n+1$ до $2n$ с соответствующими выходами блока 2. Выход конъюнкции подается на вход переключателя, запоминающего единичный сигнал. Таким образом, равенство выхода переключателя 1 эквивалентно выполнению следующих условий: во входном потоке встретилось слово $\alpha_i$ и не менее чем через $|\beta_i|$ тактов после этого во входном потоке встретилось слово $\beta_i$. Несложно заметить, что эти условия эквивалентны тому, что входное слово принадлежит языку $.*\alpha_i.*\beta_i.*$ [17].

Данная конструкция распознает язык $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$ корректно, а отсутствие экспоненциального взрыва доказано авторами работы [17] в приведенной далее теореме.

**Теорема 1** (Об отсутствии экспоненциального взрыва [17]). Для распознавания языка $\bigcup_{i=1}^{n}(.*\alpha_i.*\beta_i.*)$ представленной конструкцией требуется $O(mn\log_2(mn)+n)$ бит памяти для хранения диаграммы блока 1, и $O(n\log_2 m)$ элементов для реализации блоков 2 и 3, где $m$ — максимальная длина слов $\alpha_i$ и $\beta_i$.

Конструкция очевидным образом обобщается на классы языков с большим числом сомножителей. Структурным элементом конструкции, позволяющим решить отмеченные задачи, являются счетчики. Триггеры конструкции, отмечающие принятые слова типа $\alpha_i$, фиксируют метасостояния КА.

## Заключение

В направлении решения задачи экспоненциального взрыва через добавление специальных структурных элементов особо успешными стали некоторые идеи и алгоритмы, давшие начало направлениям исследовательской работы и практического применения в сигнатурном анализе.

Во-первых, к таким успешным решениям относится использование счетчиков. При этом находки с использованием счетчиков касаются принципиально разных регулярных языков.

Во-вторых, идея фиксировать метасостояния КА через хранение дополнительной информации о состоянии КА открыла целое направление поиска в решении проблемы экспоненциального взрыва для реализации практического поиска по сигнатурам.

В-третьих, правильная комбинация НДКА и ДКА позволяет использовать скорость ДКА и экономичность НДКА, что было продемонстрировано на практике несколькими исследовательскими группами.

В-четвертых, экономичное отношение к КА в плане его расположения к быстрой и медленной памяти позволяет учитывать эмпирический опыт накопленного вредоносного трафика в системах обнаружения вторжений и позволяет значительно сократить потребности в быстрой памяти для подобного рода сканеров на основе КА.

Приведенные решения оптимизации ресурсов часто требуют полного перебора для нахождения оптимального решения. Кроме того, предложенные решения не всегда пластичны с точки зрения изменения регулярного языка для систем обнаружения вредоносного трафика, и требуют полной пересборки КА и перенастройки структурных элементов для адаптации к изменению набора вредоносных сигнатур.

Следует отметить, что решение проблемы экспоненциального взрыва для некоторых классов регулярных языков имеет существенную практическую пользу при внедрении предложенных математических решений в работающие системы анализа трафика на вредоносность.

*Автор выражает благодарность А. В. Галатенко за комментарии и добавления, позволившие сделать работу сильнее.*

### Список литературы

1. **Кудрявцев В. Б., Алешин С. В., Подколзин А. С.** Введение в теорию автоматов, Москва: Наука, 1985, 320 с.

2. **Кудрявцев В. Б., Гасанов Э. Э. Подколзин А. С.** Основы теории интеллектуальных систем, М.: МАКС Пресс, 2016. 612 с.

3. **Документация** для Perl-совместимых регулярных выражений. URL://http://perldoc.perl.org/perlre.html

4. **Хопкрофт Дж., Мотвани Р., Ульман Дж.** Введение в теорию автоматов, языков и вычислений, М.: Вильямс, 2017. 528 с.

5. **Rabin M. O., Scott D.** Finite automata and their decision problems // IBM J. Research and Development. 1959. Vol. 3, No. 2. P. 115—125.

6. **Лупанов О. Б.** О сравнении двух типов конечных источников // Проблемы кибернетики. 1963. № 9. С. 321—326.

7. **Александров Д. Е.** Об оценках мощности некоторых классов регулярных языков // Дискретная математика. 2015. Том 27, № 2. С. 3—21.

8. **Александров Д. Е.** Об уменьшении автоматной сложности за счет расширения регулярных языков // Программная инженерия. 2014. № 11. С. 26—34.

9. **Бернадотт А.** Модификация конечного автомата через применение алгоритмов сжатия // Интеллектуальные системы. Теория и приложения. 2020. Том 24, № 3. С. 25—41.
10. **Kumar S., Chandrasekaran B., Turner J., Varghese G.** Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia // ANCS '07 Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems. 2008. P. 155—164. DOI: 10.1145/1323548.1323574.
11. **Antonatos S., Anagnostakis K., Markatos E.** Generating realistic workloads for network intrusion detection systems // Proceedings of the 4th international workshop on Software and performance, ACM SIGSOFT Software Engineering Notes. 2004. Vol. 29, No. 1. P. 207—215. DOI: 10.1145/974044.974078.
12. **Zhang X., Liu H., Wang B., Huang J., Han X.** Generating realistic network traffic and interactive application workloads using container technology // Conference: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), 2017. P. 1021— 1025. DOI: 10.1109/ICCSN.2017.8230265.
13. **Liu A. X., Norige E., Kumar S.** A few bits are enough — ASIC friendly regular expression matching for high speed network security systems // 21st IEEE Int. Conf. Netw. Protocols (ICNP). 2013. P. 1—10.
14. **Smith R., Estan C., Jha S.** XFA: Faster Signature Matching with Extended Automata // IEEE Symposium on Security and Privacy. 2008. P. 187—201.
15. **Liu C., Wu J.** Fast Deep Packet Inspection with a Dual Finite Automata // IEEE Transactions on Computers. 2013. Vol. 62, No. 2. P. 310—321. DOI: 10.1109/TC.2011.231.
16. **Becchi M., Crowley P.** Extending finite automata to efficiently match Perl-compatible regular expressions // CoNEXT. ACM, 2008. P. 25—37.
17. **Бернадотт А., Галатенко А. В.** Аппаратная конструкция для решения проблемы экспоненциального взрыва для одного класса регулярных языков // Интеллектуальные системы. Теория и приложения. 2019. Том 23, № 4. С. 27—38.

# Structural Modification of the Finite State Machine to Solve the Exponential Explosion Problem

**Bernadotte A.,** bernadotte.alexandra@intsys.msu.ru,
Department of Information Technologies and Computer Sciences, National University of Science and Technology MISIS (NUST MISIS), Moscow, 119049, Russia Federation,
Faculty of Mechanics and Mathematics, Moscow State University, Moscow, 119991, Russia Federation

*Corresponding author:*

**Alexandra Bernadotte,** Associate Professor,
Department of Information Technologies and Computer Sciences, National University of Science and Technology MISIS (NUST MISIS), Moscow, 119049, Russian Federation,
Faculty of Mechanics and Mathematics, Moscow State University, Moscow, 119991, Russian Federation
E-mail: bernadotte.alexandra@intsys.msu.ru

*Received on July 07, 2022*
*Accepted on August 10, 2022*

*In many modern applications, such as intrusion detection and prevention systems, expert knowledge can be formalized in the form of regular expressions. After this formalization, a finite automaton checks whether a word belongs to a regular language.*

*Deterministic finite automata have an optimal time complexity, but the number of automaton states (space complexity) can grow exponentially with the length of the regular expression. At the same time, the time complexity is the main disadvantage of non-deterministic finite automata. Therefore, reducing spatial complexity while maintaining low time complexity is highly relevant. In applied problems of regular language recognition, a significant problem is the problem of the exponentially growing number of states of the recognizing deterministic finite automaton depending on the length of regular expressions of the recognized language — the exponential explosion problem.*

*There are the following three main approaches to solving this problem using finite automata: 1) restriction on signatures given by experts; 2) regular language modification — this approach assumes the appearance in the solution of a practical problem of recognizing errors of the first and second kind; 3) finite automata modification without recognizing regular language changing. The third approach can be implemented as a finite automata modification through compression algorithms and particular structural elements.*

*The paper presents a review of modern solutions, the main idea of which is the transition from an abstract finite automaton, represented by a table-specified function, to a structural automaton that combines the abstract part stored in the memory and various structural elements such as bit arrays and counters.*

*Some ideas and algorithms have become especially successful when solving the exponential explosion problem by adding special structural elements.*

*First, such successful solutions include the use of counters. Second, the idea of storing additional information about the state machine. Third, the combination of non-deterministic and deterministic finite automata. Fourth, the*

*economic attitude to the state machine regarding its location to fast and slow memory allows us to consider the empirical experience of accumulated malicious traffic in intrusion detection systems.*

***Keywords:*** *finite state machine, exponential explosion problem, network intrusion detection systems, structural complexity, complexity reduction, regular languages, regular expressions*

*For citation:*

**Bernadotte A.** Structural Modification of the Finite State Machine to Solve the Exponential Explosion Problem, *Programmnaya Ingeneria,* 2022, vol. 13, no. 9, pp. 449—461.

DOI: 10.17587/prin. 13.449-461

### References

1. **Kudryavtsev V. B., Aleshin S. V., Podkolzin A. S**. *Introduction to Automata Theory*, Moscow, Nauka, 1985, 320 p. (in Russian).

2. **Kudryavtsev V. B., Gasanov E. E., Podkolzin A. S.** *Fundamentals of the theory of intelligent systems*, Moscow, MAKS Press, 2016, 612 p. (in Russian).

3. **Documentation** for Perl-compatible regular expressions, available at: http://perldoc.perl.org/perlre.html

4. **Hopcroft J., Motwani R., Ulman J.** *Introduction to the Theory of Automata, Languages and Computing*, Moscow, Williams, 2017, 528 p. (in Russian).

5. **Rabin M. O., Scott D.** Finite automata and their decision problems*, IBM J. Research and Development*, 1959, vol. 3, no. 2, pp. 115—125 (in Russian).

6. **Lupanov O. B.** Comparison of two types of finite sources, *Problems of Cybernetics*, 1963, vol. 9, pp. 321—326 (in Russian).

7. **Aleksandrov D. E.** Estimates for the cardinality of some classes of regular languages*, Discrete Mathematics*, 2015, vol. 27, no. 2, pp. 3—21 (in Russian).

8. **Aleksandrov D. E.** On the reduction of automaton complexity by extending regular languages, *Programmnaya Ingeneria*, 2014, no. 11, pp. 26—34 (in Russian).

9. **Bernadotte A.** Modification of a finite automaton through the use of compression algorithms, *Intelligent systems. Theory and Applications*, 2020, vol. 24, no. 3, pp. 25—41 (in Russian).

10. **Kumar S., Chandrasekaran B., Turner J., Varghese G.** Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia, *ANCS'07 Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, 2008, pp. 155—164. DOI: 10.1145/1323548.1323574.

11. **Antonatos S., Anagnostakis K., Markatos E.** Generating realistic workloads for network intrusion detection systems, *Proceedings of the 4th international workshop on Software and performance, ACM SIGSOFT Software Engineering Notes*, 2004, vol. 29, no. 1, pp. 207—215. DOI: 10.1145/974044.974078.

12. **Zhang X., Liu H., Wang B., Huang J., Han X.** Generating realistic network traffic and interactive application workloads using container technology, *Conference: 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, 2017, pp. 1021—1025. DOI: 10.1109/ICCSN.2017.8230265.

13. **Liu A. X., Norige E., Kumar S.** A few bits are enough — ASIC friendly regular expression matching for high speed network security systems, *21st IEEE Int. Conf. Netw. Protocols (ICNP)*, 2013, pp. 1—10.

14. **Smith R., Estan C., Jha S.** XFA: Faster Signature Matching with Extended Automata, *IEEE Symposium on Security and Privacy*, 2008, pp. 187—201.

15. **Liu C., Wu J.** Fast Deep Packet Inspection with a Dual Finite Automata, *IEEE Transactions on Computers*, vol. 62, no. 2, 2013, pp. 310—321. DOI: 10.1109/TC.2011.231.

16. **Becchi M., Crowley P.** Extending finite automata to efficiently match Perl-compatible regular expressions, *CoNEXT*. ACM, 2008, pp. 25—37.

17. **Bernadotte A., Galatenko A. V.** A hardware design for solving the exponential explosion problem for a class of regular languages, *Intelligent Systems. Theory and Applications*, 2019, vol. 23, no. 4, pp. 27—38 (in Russian).

УДК 629.113
DOI: 10.17587/prin.13.462-468

**С. М. Зуев,** канд. физ.-мат. наук, доц. кафедры, sergei_zuev@mail.ru, МИРЭА — Российский технологический университет, **Д. О. Варламов**, ст. препод., varlamovd@mail.ru, **У. И. Акрамов**, студент, ulugbekakramov97@gmail.com, **В. В. Кукса,** студент, vetal11@yandex.ru, Московский политехнический университет

# Программно реализованная модель устройства контроля параметров автоматической коробки переключения передач автомобиля

*Представлено описание программно реализованной модели устройства для контроля параметров автоматической коробки переключения передач (АКПП) автомобиля, которое позволяет в реальном времени отслеживать такие параметры, как температура масла АКПП, частота вращения валов, исправность как самих датчиков давления, так и клапанов, которые отвечают за нарастание давления и удерживание его в нужных пределах для задействования тех или иных датчиков давления. Благодаря своей простоте это устройство может широко применяться как в автосервисе, так и в производстве. Кроме того, устройство позволяет определить причину неисправности АКПП на ранней стадии и предотвратить дорогостоящий ремонт.*

***Ключевые слова:*** *автоматическая коробка переключения передач (АКПП), микроконтроллер Atmega32, датчики Холла, терморезистивные датчики, температура масла, частота вращения валов*

## Введение

Хорошо известно, что коробка переключения передач (КПП) в автомобилях устанавливается между двигателем и приводом в целях переноса крутящего момента с двигателя на приводной вал и для изменения его скорости вращения [1]. Она может быть выполнена с ручным или с электронным управлением. У коробок первого типа, как правило, сцепление однодисковое, у коробок второго типа оно может быть как однодисковым, так и двухдисковым. Принцип работы КПП основан на передаче крутящего момента от двигателя к приводу, в котором он преобразуется в движение колес.

Если бы двигатель имел жесткое сцепление с приводным валом, то пришлось бы при остановке глушить двигатель, нельзя было бы плавно трогаться с места [2]. Использование КПП необходимо, так как двигатель производит свою максимальную мощность только на узком диапазоне скоростей и при наличии только одной передачи это означало бы, что автомобиль имел бы ограниченную максимальную скорость или очень плохую производительность на низких скоростях [3].

Сегодня автоматическая коробка переключения передач (далее — АКПП) является наиболее востребованным типом трансмиссии по целому ряду причин [4]. Прежде всего, данный тип КПП значительно упрощает процесс управления автомобилем, езда становится более комфортной и безопасной, так как водитель не отвлекается на переключение передач, исключены ошибки при выборе передачи и т. п. Также АКПП является более сложным агрегатом по сравнению с механической КПП. Вместе с тем известно, что чем сложнее устройство, тем больше вероятность возникновения серьезных поломок [5]. Конечно, как и любая другая система, АКПП требует внимания и регулярного обслуживания. Одним из частых проблемных вопросов, с которым сталкиваются автовладельцы, является появление вибрации в АКПП. Причин может быть несколько: изношенные детали; неисправность гидроблока; выход из строя гидронасоса; некорректная работа фрикционов. Поэтому анализ работоспособности АКПП является важной задачей в современном машиностроении.

## 1. Постановка задачи

При разработке и дальнейшем обслуживании АКПП используются специальные стенды. Однако основным неудобством их использования является то обстоятельство, что некоторые параметры невозможно отслеживать. К числу таких параметров относятся: частота вращения на входном и выходном валах; температура масла непосредственно с датчика; исправность датчиков давления [6].

Принимая во внимание отмеченные выше сложности, была поставлена задача разработать и сконструировать дополнительное устройство, с помощью которого можно было бы контролировать эти параметры в совместной работе со стендом AXILINE 97000 ECRH. Данное устройство разрабатывалось для дополнительного тестирования электронной части АКПП, так как бывшие в использовании датчики, электрические цепи и электромагнитные клапаны имеют свойство выхюдить из строя и имеющийся стенд AXILINE 97000 ECRH не позволяет их тестировать и выявлять бракованные датчики и провода (косы) [7, 8].

На рис. 1 (см. вторую сторону обложки) приведена электрическая схема разработанного авторами устройства. На основе разработанной электрической схемы был собран опытный образец, который включает в себя перечисленные далее основные элементы.

За управление устройством отвечает 8-разрядный микроконтроллер (МК) Atmega32A, построенный на основе архитектуры AVR RISC [9]. Такой МК тактируется от внешнего кварцевого резонатора на 8 МГц. Для отсечения гармоник высшего порядка к кварцевому резонатору подключаются конденсаторы C4 и C5. МК питается от внешнего адаптера на 5 В. Для фильтрации высокочастотных помех по питанию МК служит конденсатор C3, а конденсатор C6 защищает от возможных колебаний напряжения питания. Для уменьшения наводок и помех по питанию встроенного в МК аналого-цифрового преобразователя (АЦП) применяется LC-фильтр на индуктивности L1 и конденсаторе C2 [10]. Внутрисхемное программирование этого МК осуществляется через стандартный 6-пиновый разъем ISP [11].

Резисторы R6 и R7 соединяются последовательно к датчику Холла входного вала АКПП, а резисторы R9 и R8 — соответственно к датчику Холла выходного вала АКПП. Сигналы с этих датчиков (IN_shaft и OUT_shaft) поступают на схему преобразования сигналов, построенную на основе микросхемы LM358N, в которой находятся два операционных усилителя (ОУ). Датчики Холла и ОУ микросхемы LM358N питаются напряжением 12 В от повышающего стабилизатора напряжения XL6009. На инвертирующие входы этих ОУ с потенциометра RP1 подается напряжение 10,5 В. Если на вход IN_shaft первого ОУ подается сигнал с напряжением, превышающим 10,5 В, то на выходе 1OUT будет сигнал с амплитудой, близкой к 11 В, если на вход IN_shaft поступает сигнал с амплитудой менее 10,5 В, то на выходе 1OUT будет 0 В. То же будет справедливо для входа OUT_shaft и выхода 2OUT второго ОУ. Резисторы R5 и R4 образуют делитель напряжения, который в 3 раза уменьшает амплитуду сигнала с 1OUT первого ОУ, чтобы его можно было подать на вход INT0 (вход внешних прерываний) МК. Аналогично, сигнал с 2OUT второго ОУ преобразуется в INT1 (вход внешних прерываний) МК, уменьшаясь в 3 раза на делителе напряжения, образованного резисторами R2 и R3.

Результат измерения частоты вращения входного и выходного валов АКПП в двоичном коде передается по интерфейсу SPI на микросхему управления сегментными индикаторами MAX7219 [12, 13]. Такая микросхема, используя метод динамической индикации, выводит на верхний индикатор частоту вращения входного вала, а на нижний индикатор — частоту вращения выходного вала АКПП [14].

Резистор R10 образует делитель напряжения с терморезистивным датчиком температуры АКПП. Сигнал "Temper" с датчика поступает на вход PA0 МК, который настроен как аналоговый вход (к нему подключен внутренний АЦП).

Результат измерения напряжения на датчике температуры пересчитывается в температуру и выводится на знакосинтезирующий ЖК-дисплей в полубайтном режиме.

Сигналы с трех мембранных датчиков давления поступают напрямую на индикаторные светодиоды LED1—LED3 через токоограничивающие резисторы R11—R13.

Написание кода для МК разработанного устройства осуществлялось в интегрированной среде разработки Microchip Studio. Рабочее окно программы представлено на рис. 2 (см. вторую сторону обложки).

Рассмотрим настройку 16-битного таймера-счетчика T1 для измерения частоты вращения входного и выходного валов АКПП (см. таблицу). Основная модель АКПП, для которых предназначено данное устройство, — это достаточно популярная 6-ступенчатая АКПП U660E производства компании Aisin, устанавливаемая в автомобили как японского, так и европейского и американского производства.

**Расчет значения входного и выходного валов**

| Вал АКПП | $n$ | $v$, об/мин | $\tau_п$, мкс |
|---|---|---|---|
| Входной | 40 | 50 | 30 000 |
| | | 6000 | 250 |
| Выходной | 44 | 50 | 27 272 |
| | | 6000 | 227 |

Входной вал такой АКПП имеет 40 зубцов, выходной — 44. Максимальная частота вращения, до которой требуется проводить измерения, составляет 6000 об/мин, а минимальная — 50 об/мин.

Время $\tau_п$, за которое мимо датчика Холла проходит один зубец вала, определяется по формуле:

$$\tau_п = \frac{T_м}{\nu n}, \quad (1)$$

где $T_м = 60$ с — значение для пересчета в об/с; $\nu$ — частота вращения вала, об/мин; $n$ — число зубцов вала.

Для измерения периодов времени для входного и выходного валов применим 16-битный таймер/счетчик микроконтроллера Atmega32. Он будет считать мкс между приходами зубцов валов АКПП. Поскольку счетчик 16-битный, время в микросекундах, которое он сможет измерить при времени такта в 1 мкс, составит 0,065535 с. Следовательно, минимальная частота вращения, которую он сможет определить, составит

$$\nu = \frac{T_м}{\tau_п n} = \frac{60}{0,065535 \cdot 40} = 22,9 \text{ об/мин}. \quad (2)$$

В случае, если частота составит меньшее значение, это приведет к срабатыванию прерывания по переполнению счетчика, и на индикаторах будут отображаться тире ("----").

Используем режим работы таймера/счетчика Normal — он установлен в нем по умолчанию. Такт таймера/счетчика clkT1 определяется по формуле

$$clk_{T1} = \frac{f_{CPU}}{N}, \quad (3)$$

где $f_{CPU}$ — частота тактирования ядра МК, Гц (в разрабатываемом устройстве оно составляет 8 МГц); $N$ — коэффициент предделителя таймера/счетчика Т1 МК.

Выбор источника тактирования ядра МК проводится на закладке "Fuses" (рис. 3, см. третью сторону обложки) в программе Microchip Studio при подключении программатора к разъему программирования ISP. Здесь выбран внешний высокочастотный кристалл/резонатор, время выхода в рабочий режим, соответствующий 16 тактовым циклам, и задержка запуска 0 мс.

Чтобы время такта таймера составило 1 мкс, частота на выходе предделителя таймера должна составить 1 МГц. Для этого выбираем тактовый сигнал от встроенного генератора с предделителем на 8. Это выполняется путем записи логической единицы в бит CS11 регистра TCCR1B:

$$\text{TCCR1B}| = (1<<\text{CS11}).$$

Чтобы зафиксировать моменты прихода сигналов от датчика Холла, необходимо разрешить внешние прерывания для входов МК INT0 и INT1 (куда поступают сигналы от датчиков, преобразованные компаратором) по перепаду с 0 на 1:

$$\text{GICR}| = (1<<\text{INT0}) |(1<<\text{INT1});$$

$$\text{MCUCR}| = (1<<\text{ISC01})|(1<<\text{ISC11}).$$

Чтобы определять актуальное значение скорости вращения валов АКПП, необходимо измеренное (в мкс) значение в счетном регистре TCNT1, переведенное в секунды из микросекунд, подставить в формулу (2).

Рассмотрим построение зависимости напряжения от температуры для терморезистивного датчика.

В технических описаниях датчиков температуры обычно предоставляются табличные зависимости их сопротивления от температуры.

Это позволяет провести оперативное измерение сопротивления терморезистивных датчиков мультиметром в режиме омметра, что используется при проведении их диагностики. Графическое представление характеристики изображено на рис. 4. Судя по представленной зависимости, данный датчик является термистором, так как имеет отрицательный температурный коэффициент (с ростом температуры сопротивление падает).

Напряжение, измеряемое МК, определяется по формуле

$$U_{изм} = \frac{U_п R_t}{R_п + R_t}, \quad (4)$$

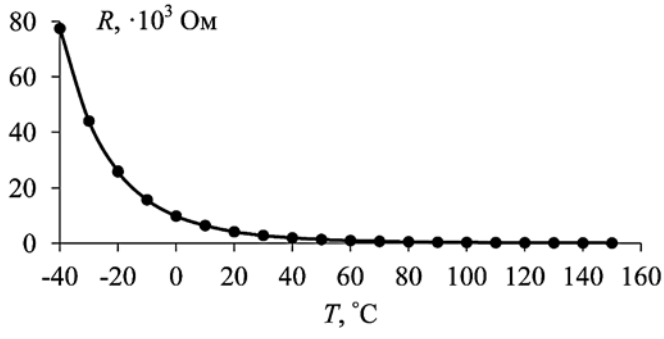


**Рис. 4. Графическое представление зависимости сопротивления от температуры для терморезистивного датчика в логарифмическом масштабе**

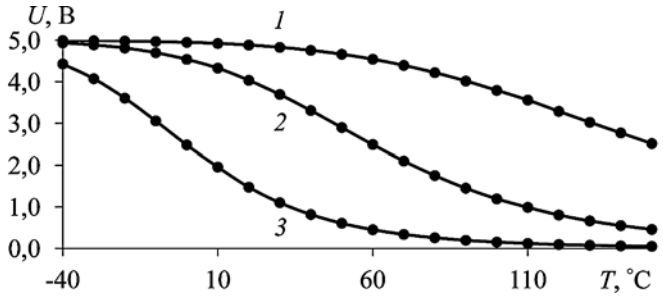


**Рис. 5. Зависимости измеряемого микроконтроллером напряжения от температуры:**
*1* — $R_п$ = 100 Ом; *2* — $R_п$ = 1 кОм; *3* — $R_п$ = 10 кОм

где $U_п$ — напряжение питания датчика (5 В); $R_п$ — постоянное сопротивление делителя напряжения; $R_t$ — сопротивление термистора.

Проведем расчет зависимости измеряемого напряжения от сопротивления датчика для трех разных значений $R_п$. Рассчитанные значения представлены графически на рис. 5.

При сопротивлении $R_п$ = 100 Ом зависимость измеряемого напряжения от температуры занимает только половину от измеряемого диапазона от 0 до 5 В. При сопротивлении $R_п$ = 10 кОм на температурах более 80 °С, на 10 °С изменения температуры изменение напряжения составит менее 50 мВ, что скажется на точности измерения, особенно учитывая то, что они будут проводится АЦП в 8-битном режиме. При сопротивлении $R_п$ = 1 кОм на 10 °С изменения температуры изменение напряжения составит не менее 100 мВ даже на граничных участках характеристики. Поэтому выбираем характеристику при $R_п$ = 1 кОм.

Поскольку МК определяет температуру по напряжению на терморезистивном датчике, то необходимо перестроить зависимость на рис. 5 при $R_п$ = 1 кОм так, чтобы по оси абсцисс было отложено напряжение, а по оси ординат — температура. Такая зависимость представлена на рис. 6.

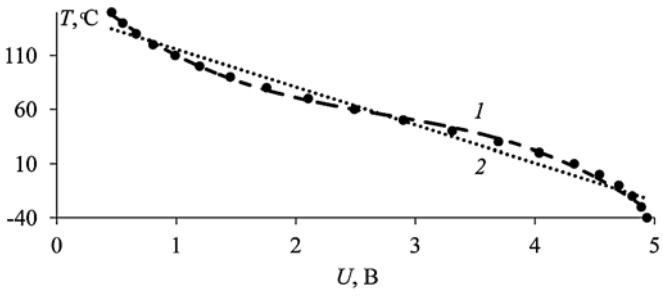


**Рис. 6. Зависимость температуры от измеряемого микроконтроллером напряжения с полиномиальной (*1*) и линейной (*2*) линией тренда. Точками показаны экспериментальные значения *T*:**
*1* — $y = -4{,}182x^3 + 34{,}259x^2 - 112{,}79x + 192{,}8$; 2 − $y = -35{,}057x +$ + 150,72

Полученная зависимость является нелинейной. Для того чтобы МК мог пересчитывать значение напряжения в температуру, в программе Microsort Excel были построены линии тренда (полиномиальная *1* и линейная *2* на рис. 6). Данные линии позволяют подобрать математическую функцию, наиболее оптимально описывающую полученную зависимость. В результате расчета получается таблица с параметрами и результатами моделирования. Линейная функция достаточно проста, но не точно описывает реальную функцию. Полиномиальная функция избыточно сложная для поставленной задачи, с которой будет работать 8-битный МК. При выполнении этой задачи на одном МК нельзя использовать память, чтобы записывать в нее полиномы. Кроме того, при записи полиномиальной функции в память будет потеряно целочисленное значение, которое нужно будет подставлять в качестве операнда. В связи с этим наиболее оптимальным вариантом будут являться разбиение линии на три участка и линеаризация их по отдельности с получением математических уравнений (рис. 7). При этом возможно использование метода последовательных приближений, но по мере приближения к заданной точности лучше воспользоваться методом уточненного максимума. В результате линеаризации получается система линейных алгебраических уравнений (СЛАУ), которая позволяет определить положение и размеры каждого отрезка линии.

Если рассматривать график в линейной шкале, то можно сразу увидеть три важные вещи: во-первых, при увеличении напряжения с точки зрения аналоговой логики наблюдается увеличение потребляемого тока, во-вторых, при превышении некоторого порога (например, 5 В) на выходе микросхемы появляется сигнал логической 1. В-третьих, линейная аппроксимация хорошо работает

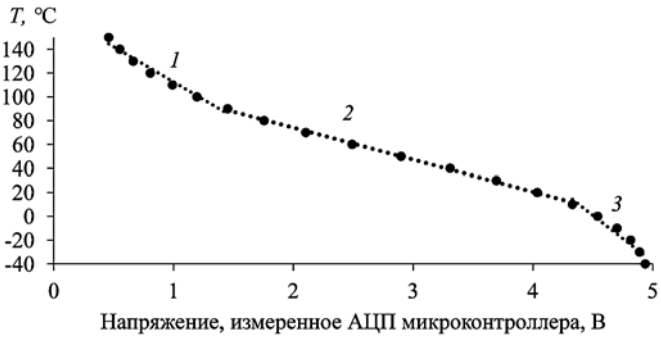


**Рис. 7. Зависимость температуры от измеряемого микроконтроллером напряжения с тремя линеаризованными участками:**
*1* — $y = -59{,}221x + 171{,}74$; *2* — $y = -26{,}802x + 127{,}62$; *3* — $y = -78{,}249x + 352{,}87$

для аналоговых сигналов и не очень хорошо, а иногда вовсе не работает, для цифрового сигнала. Так что на практике часто приходится пользоваться сложной аппроксимацией (например, сплайном). В-четвертых, часто для аппроксимации используются не один полином, а несколько. Например, для трех точек — четыре и т. д. Чем больше полиномов участвует в аппроксимации, тем лучше она "сглаживается" и тем меньше влияние разброса параметров отдельных полиномов. В общем случае — чем больше точек взято в полиномном уравнении аппроксимирующего функционала, тем хуже качество аппроксимации.

Зависимость температуры от измеряемого МК напряжения с тремя линеаризованными участками представлена на рис. 7.

По мере увеличения напряжения температура падает, и соответственно, сопротивление также падает.

## 2. Моделирование работы устройства с помощью программы Proteus

Далее рассмотрим моделирование работы устройства. Для правильной настройки работы устройства оно было смоделировано с использованием программы Proteus. Данный симулятор является универсальным средством, позволяющим моделировать работу реального устройства. Среди основных функций можно выделить:

- моделирование работы устройства;
- вывод результатов;
- запуск процедуры;
- получение подробных отчетов о работе устройства.

Модель устройства в программе Proteus представлена на рис. 8, см. третью сторону обложки.

Осциллограммы сигналов, поступающих на входы INT0 и INT1, с диаграммой, поясняющей принцип измерения устройством частоты входных сигналов, представлены на рис. 9, см. четвертую сторону обложки.

Измерение частоты проводится описанным далее способом.

- Для измерения частоты вращения входного вала включается внешнее прерывание INT0 (входной вал) по переднему фронту. Как только на вход INT0 поступит импульс, сработает прерывание, которое запустит счетчик T1, длительность каждого такта которого составляет 1 мкс. По приходу второго импульса происходит выключение внешнего прерывания INT0, остановка счетчика и запись в переменную результата измерения длительности периода сигнала входного вала значения счетного регистра TCNT1.

- Для измерения частоты вращения выходного вала включается внешнее прерывание INT1 (выходной вал) по переднему фронту. Как только на вход INT1 поступит импульс, сработает прерывание, которое запустит счетчик T1. По приходу второго импульса происходит выключение внешнего прерывания INT1, остановка счетчика и запись в переменную результата измерения длительности периода сигнала выходного вала значения счетного регистра TCNT1.

В примере на рис. 9 (см. четвертую сторону обложки) при измерении частоты вращения входного вала в счетном регистре TCNT1 значение составило 855, следовательно, частота вращения (в об/мин) для входного вала составляет $60/(0{,}000855\times40) = 1754$ об/мин. Для выходного вала значение в TCNT1 составило 1425, следовательно, частота вращения выходного вала $60/(0{,}001425\times44) = 956$ об/мин.

Для работы с МК основными источниками получения информации являются подсистемы Reference Manual, а также Data Sheet. В них даны характеристики МК, назначение выводов, электрические параметры, описание управляющих регистров, описание периферийных подсистем. АЦП МК измеряет напряжение на входе PA0/ADC1 и пересчитывает значение напряжения в температуру, которая отображается на знакосинтезирующем ЖК-дисплее.

Пересчет осуществляется по одной из трех формул, в зависимости от измеренного напряжения:

$$\text{если } U_{\text{изм}} < 1{,}1 \text{ В, то } T = 171{,}4 - 59{,}9221U_{\text{изм}};$$

$$\text{если } 1{,}1 \text{ В} \leqslant U_{\text{изм}} \leqslant 4{,}3 \text{ В, то } T = 127{,}62 - 26{,}802U_{\text{изм}};$$

$$\text{если } U_{\text{изм}} > 4{,}3, \text{ то } T = 352{,}87 - 78{,}249U_{\text{изм}}.$$

В примере на рис. 8 (см. третью сторону обложки) показано, что при напряжении 2,55 В на потенциометре *1* на ЖК-дисплее *2* отображается температура 60 °С.

Для проверки зависимостей применим вторую из представленных формул: $T = 127{,}62 - 26{,}802\times2{,}55 \approx 60$ — верно.

Таким образом, на основе проведенной работы было сконструировано устройство для контроля параметров АКПП автомобиля, которое предназначено для повышения точности диагностики данного типа устройств (рис. 10, см. четвертую сторону обложки).

## Заключение

Описана программно реализованная модель устройства для контроля параметров АКПП автомобиля. Данная модель позволяет не только в реальном времени отслеживать критерии, которые

предъявляются для оценки качества функционирования устройства (температура масла АКПП; частота вращения валов; исправность датчиков давления и клапанов, которые отвечают за нарастание давления и удерживания его в нужных пределах для задействования тех или иных датчиков давления), но и применять полученные схемные результаты для анализа похожих систем. Рассмотренное устройство благодаря своей простоте может иметь широкий спектр применения как в автосервисе, так и в производстве. Кроме того, оно позволяет определить причину неисправности АКПП на ранней стадии и предотвратить дорогостоящий ремонт, что дает экономическую выгоду от его использования. Спроектированное устройство работает исправно в реальном времени, отображая данные частоты вращения входного и выходного валов на семисегментных индикаторах. На дисплее выводятся также данные о состоянии температуры терморезистивных датчиков. Устройство имеет малую массу и габаритные размеры, удобную индикацию, универсальный разъем подключения, может быть запитано от любого блока питания постоянного тока с напряжением 5 В и имеет погрешность измерения менее 0,5 % на интервале от 50 до 6000 об/мин. Все перечисленные показатели делают устройство универсальным и простым в использовании. А высокий класс точности позволяет проводить не только ремонтно-восстановительные работы, но и работы, связанные с разработкой и проектированием новых АКПП, в том числе для автотранспорта с гибридными силовыми установками.

Таким образом, представленное программно реализованное устройство контроля параметров АКПП является актуальным и недорогим решением, что позволяет не разрабатывать и не использовать дорогостоящие системы натурного исследования.

### Список литературы


1. **Salamandra K., Tyves L.** Modeling of cyclic gearshifts in automatic transmission for vehicles // Engineering for Rural Development. 17th International Scientific Conference. 2018. P. 2071—2078. DOI: 10.22616/ERDev2018.17.N331.

2. **Blokhin A., Barakhtanov L., Fadeev E., Lubichev P.** Research of robotized manual transmissions for all-terrain vehicles // ARPN Journal of Engineering and Applied Sciences. 2017. Vol. 12, No. 1. P. 20—32.

3. **Иванов А. С., Лянденбурский В. В., Фахрутдинов И. И., Экимов П. М., Иванов В. А., Кухмазов К. З.** Контроль технического состояния автоматической коробки передач // Нива Поволжья. 2019. № 4 (53). С. 121—128. DOI: 10.36461/NP.2019.52.3.018.

4. **Abyzov O. V., Dobretsov R. Y., Sidorov A. A., Galyshev Y. V., Krasilnikov A. A., Uvakina D. V.** Stationary text complex for vehicle and tractor gearboxes // International Review of Mechanical Engineering. 2020. Vol. 14, No. 2. P. 127—132. DOI: 10.15866/ireme.v14i2.18267.

5. **Белабенко Д. С.** Типы исполнительных механизмов современных гидромеханических передач отечественных и зарубежных производителей // Механика машин, механизмов и материалов. 2016. № 3 (36). С. 34—42.

6. **Liu J., Li Q., Yu L., Zeng Q.** Synthesis of multi-row and multi-speed planetary gear mechanism for automatic transmission // Mechanism and Machine Theory. 2018. Vol. 128. P. 616—627. DOI: 10.1016/J.MECHMACHTHEORY.2018.07.007.

7. **Экимов П. М., Лянденбурский В. В., Борисов Н. Б.** Контроль неисправностей автоматической коробки передач // Бюллетень транспортной информации. 2017. № 10 (268). С. 16—19.

8. **Алехин В. А.** Проектирование электронных систем с использованием SYSTEMC и SYSTEMC-AMS // Российский технологический журнал. 2020. Том 8, № 4 (36). С. 79—95. DOI: 10.32362/2500-316X-2020-8-4-79-95.

9. **Анисимов И. А., Хлопотов Р. А.** Обоснование актуальности исследования зависимости расхода топлива двигателя на режиме холостого хода от режима работы АКПП и температурных условий // Научно-технический вестник Поволжья. 2013. № 6. С. 120—122.

10. **Зуев С. М., Варламов Д. О., Лавриков А. А., Малеев Р. А., Шматков Ю. М.** Краткий толковый русско-английский терминологический словарь-справочник. М.: ИНФРА-М, 2021. 200 с. DOI: 10.12737/1242228.

11. **Turner J. D., Austin L.** Sensors for automotive telematics // Measurement Science and Technology. 2000. Vol. 11, No. 2. P. R58—R79.

12. **Малеев Р. А., Зуев С. М., Скворцов А. А., Великий М. Д., Свинцов М. В.** Экспериментальные исследования ДВС с емкостными накопителями энергии // Вестник машиностроения. 2021. № 2. С. 32—36. DOI: 10.36652/0042-4633-2021-2-32-36.

13. **Малеев Р. А., Зуев С. М., Скворцов А. А., Лавриков А. А.** Системы электростартерного пуска с высоковольтовыми источниками тока // Проблемы машиностроения и автоматизации. 2020. № 1. С. 66—71.

14. **Малеев Р. А., Зуев С. М., Лавриков А. А., Гребенчиков Н. П.** Исследование режимов работы емкостных накопителей энергии в системах пуска автомобильных двигателей // Известия МГТУ "МАМИ". 2019. № 1 (39), С. 29—35. DOI: 10.31992/2074-0530-2019-39-1-29-35.


# A Software-Implemented Model of a Device for Monitoring the Parameters of an Automatic Gearbox of a Car


**S. M. Zuev,** sergei_zuev@mail.ru,
MIREA — Russian Technological University, Moscow, 119454, Russian Federation,
**D. O. Varlamov,** e-mail: varlamovd@mail.ru, **U. I. Akramov,** ulugbekakramov97@gmail.com,
**V. V. Kuksa**, ya.vetal11@yandex.ru,
Moscow Polytechnic University, Moscow, 432700, Russian Federation

*Corresponding author:*

**Sergei M. Zuev,** PhD, Associate Professor,
MIREA — Russian Technological University, Moscow, 119454, Russian Federation
E-mail: sergei_zuev@mail.ru

*Received on July 15, 2022*
*Accepted on July 28, 2022*

*The paper presents a description of a device for monitoring the parameters of an automatic transmission of a car, which allows you to monitor in real time such parameters as the temperature of the automatic transmission oil, the rotational speed of the shafts, the serviceability of both the pressure sensors themselves and the valves that are responsible for the increase in pressure and keeping it within the required limits to activate certain pressure sensors. This device, due to its simplicity, has a wide range of applications, both in car service and in production. In addition, the device allows you to determine the cause of the automatic transmission malfunction at an early stage and prevent costly repairs.*

***Keywords:*** *automatic gearbox (automatic transmission), Atmega32 microcontroller, Hall sensors, thermoresistive sensors, oil temperature, shaft speed*

*For citation:*

**Zuev S. M.**, **Varlamov D. O., Akramov U. I.**, **Kuksa V. V.** A Software-Implemented Model of a Device for Monitoring the Parameters of an Automatic Gearbox of a Car, *Programmnaya Ingeneria*, 2022, vol. 13, no. 9, pp. 462—468.

DOI: 10.17587/prin.13.462-468

**References**

1. **Salamandra K., Tyves L.** Modeling of cyclic gearshifts in automatic transmission for vehicles, *Engineering for Rural Development. 17th International Scientific Conference*, 2018, pp. 2071-2078. DOI: 10.22616/ERDev2018.17.N331.

2. **Blokhin A., Barakhtanov L., Fadeev E., Lubichev P.** Research of robotized manual transmissions for all-terrain vehicles, *ARPN Journal of Engineering and Applied Sciences,* 2017, vol. 12, no. 1, pp. 20—32.

3. **Ivanov A. S., Lyandenburskiy V. V., Fakhrutdinov I. I., Ekimov P. M., Ivanov V. A., Kukhmazov K. Z.** Control of Technical Condition of the Automatic Transmission, *Volga Region Farmland,* 2019, no. 4, pp. 84—89. DOI: 10.26177/VRF.2020.4.4.017.

4. **Abyzov O. V., Dobretsov R. Y., Sidorov A. A., Galyshev Y. V., Krasilnikov A. A., Uvakina D. V.** Stationary text complex for vehicle and tractor gearboxes, *International Review of Mechanical Engineering*, 2020, vol. 14, no. 2, pp. 127—132. DOI: 10.15866/ireme.v14i2.18267.

5. **Belabenko D. S.** Types of actuators of modern hydromechanical transmissions of domestic and foreign manufacturers, *Mechanics of machines, mechanisms and materials,* 2016, no. 3 (36), pp. 34—42 (in Russian).

6. **Liu J., Li Q., Yu L., Zeng Q.** Synthesis of multi-row and multi-speed planetary gear mechanism for automatic transmission, *Mechanism and Machine Theory*, 2018, vol. 128, pp. 616—627. DOI: 10.1016/J.MECHMACHTHEORY.2018.07.007.

7. **Ekimov P. M., Lyandenburgsky V. V., Borisov N. B.** Control of malfunctions of an automatic transmission, *Transport Information Bulletin*, 2017, no. 10 (268), pp. 16—19 (in Russian).

8. **Alekhine V. A.** Design of electronic systems using SYSTEMC and SYSTEMC-AMS, *Russian technological journal*, 2020, vol. 8, no. 4 (36), pp. 79—95. DOI: 10.32362/2500-316X-2020-8-4-79-95.

9. **Anisimov I. A., Khlopotov R. A.** Substantiation of the relevance of the study of the dependence of engine fuel consumption in idle mode on the automatic transmission operating mode and temperature conditions, *Scientific and technical bulletin of the Volga region,* 2013, no. 6, pp. 120—122 (in Russian).

10. **Zuev S. M., Varlamov D. O., Lavrikov A. A., Maleev R. A., Shmatkov Yu. M.** *Brief explanatory Russian-English terminological dictionary-reference book,* Moscow, INFRA-M, 2021, 200 p. DOI: 10.12737/1242228 (in Russian).

11. **Turner J. D., Austin L.** Sensors for automotive telematics. Measurement Science and Technology. 2000. V. 11. No. 2. P. R58-R79.

12. **Maleev R. A., Zuev S. M., Skvortsov A. A., Veliky M. D., Svintsov M. V.** Experimental studies of internal combustion engines with capacitive energy storage, *Bulletin of mechanical engineering*, 2021, no. 2, pp. 32—36. DOI: 10.36652/0042-4633-2021-2-32-36 (in Russian).

13. **Maleev R. A., Zuev S. M., Skvortsov A. A., Lavrikov A. A.** Electric start systems with high-voltage current sources, *Problems of mechanical engineering and automation,* 2020, no. 1, pp. 66—71 (in Russian).

14. **Maleev R. A., Zuev S. M., Lavrikov A. A.**, **Grebenchikov N. P.** Investigation of operating modes of capacitive energy storage devices in start-up systems of automobile engines, *Proceedings of MSTU "MAMI",* 2019. no. 1 (39), p. 29—35. DOI: 10.31992/2074-0530-2019-39-1-29-35 (in Russian).

**ООО "Издательство "Новые технологии".** 107076, Москва, ул. Матросская Тишина, д. 23, стр. 2
Технический редактор *Е. М. Патрушева.* Корректор *А. В. Чугунова.*

Сдано в набор 09.09.2022 г. Подписано в печать 14.10.2022 г. Формат 60×88 1/8. Заказ PI921
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**

Рисунки к статье **С. М. Зуева, Д. О. Варламова, У. И. Акрамова, В. В. Куксы**

# «ПРОГРАММНО РЕАЛИЗОВАННАЯ МОДЕЛЬ УСТРОЙСТВА КОНТРОЛЯ ПАРАМЕТРОВ АВТОМАТИЧЕСКОЙ КОРОБКИ ПЕРЕКЛЮЧЕНИЯ ПЕРЕДАЧ АВТОМОБИЛЯ»

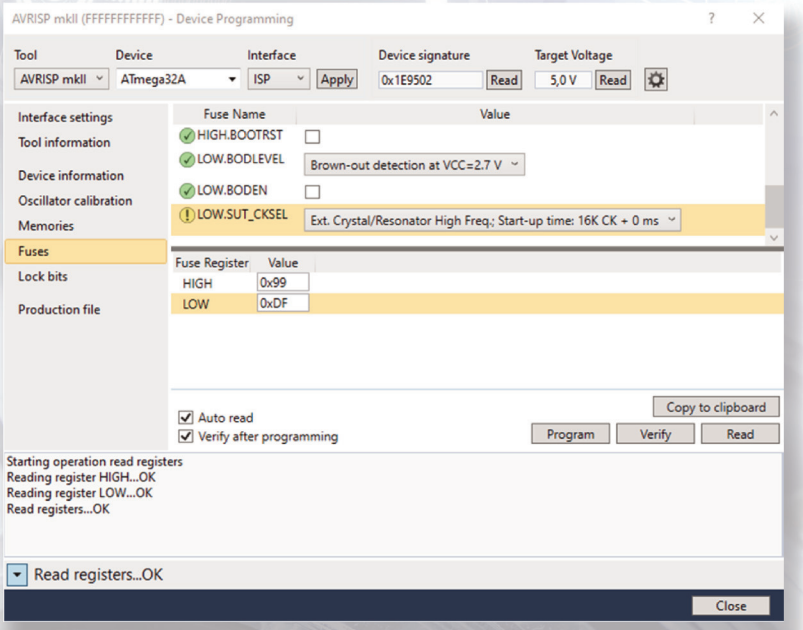


**Рис. 3. Окно установки фьюз-битов микроконтроллера**

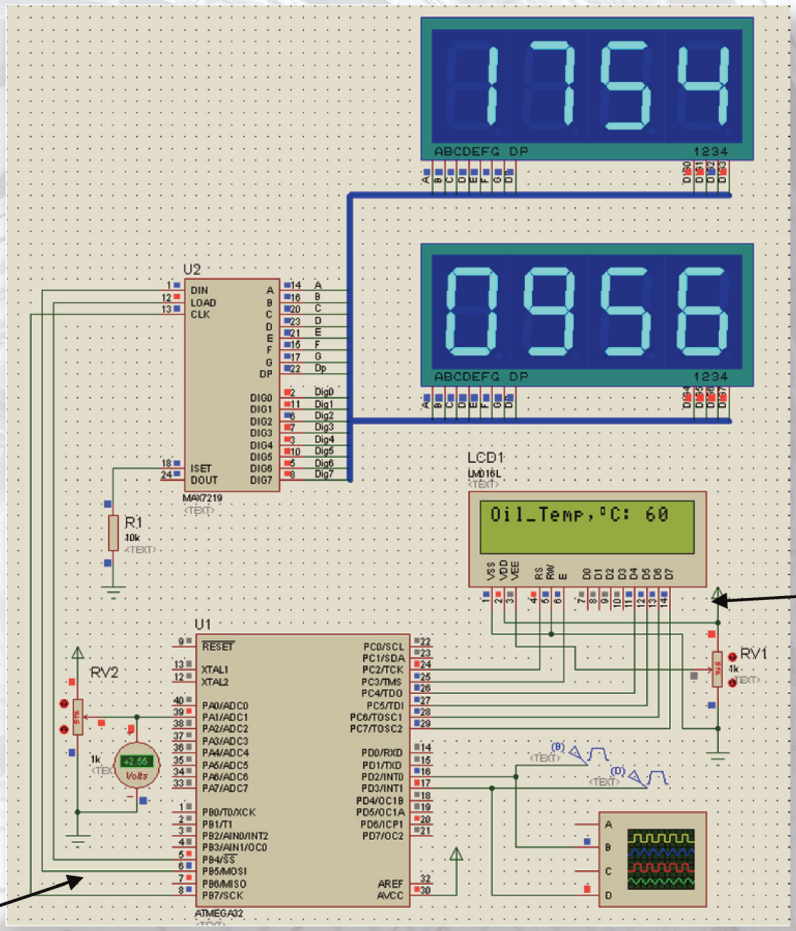


**Рис. 8. Модель разработанного устройства в программе Proteus:**
*1* – потенциометр; *2* – ЖК-дисплей

Рисунки к статье **С. М. Зуева, Д. О. Варламова, У. И. Акрамова, В. В. Куксы**

## «ПРОГРАММНО РЕАЛИЗОВАННАЯ МОДЕЛЬ УСТРОЙСТВА КОНТРОЛЯ ПАРАМЕТРОВ АВТОМАТИЧЕСКОЙ КОРОБКИ ПЕРЕКЛЮЧЕНИЯ ПЕРЕДАЧ АВТОМОБИЛЯ»

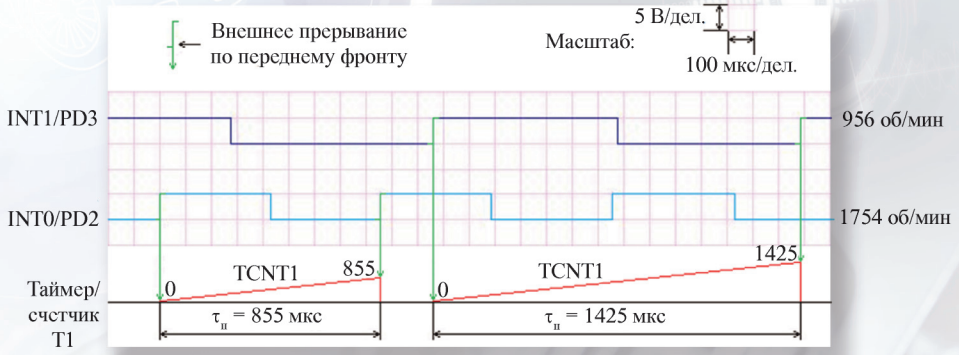


**Рис. 9. Осциллограмма входных сигналов с датчиков в программе Proteus**

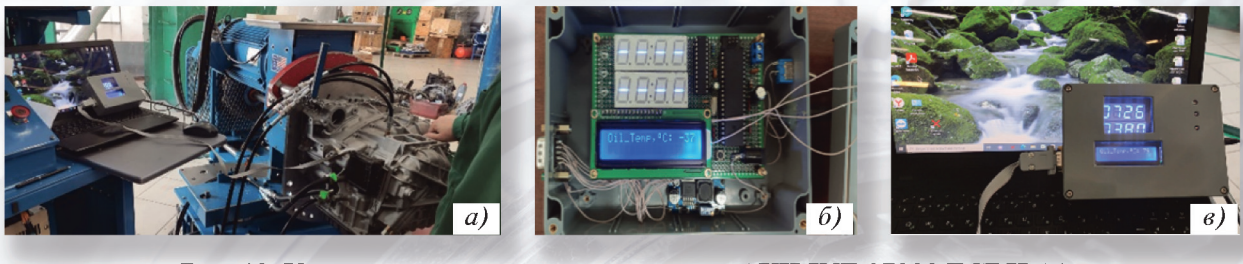


**Рис. 10. Установка на испытательном стенде AXILINE 97000 ECRH (*а*), а также внутренний вид (*б*) и внешний вид (*в*) устройства**

Рисунок к статье **Е. В. Орловой**

## «СИСТЕМНЫЙ ИНЖИНИРИНГ ЦИФРОВЫХ ДВОЙНИКОВ ОРГАНИЗАЦИОННО-ТЕХНИЧЕСКИХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА»

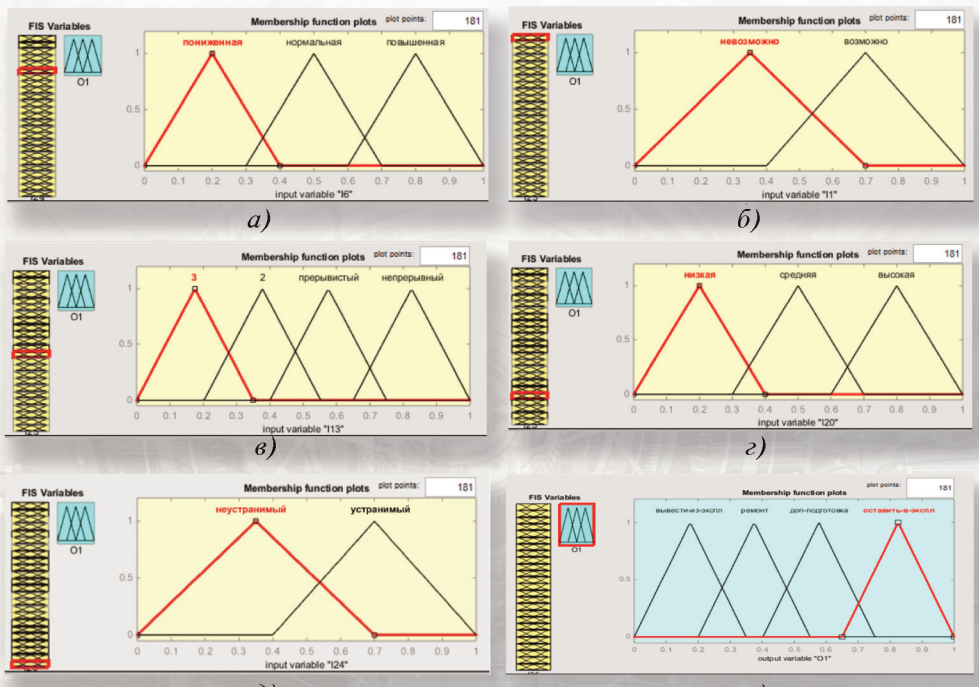


**Рис. 5. Функции принадлежности для входных и выходной переменных:**
*а* – $I_6$; *б* – $I_1$; *в* – $I_{13}$; *г* – $I_{20}$; *д* – $I_{24}$; *е* – $O_1$

ISSN 2220-3397. Программная инженерия. 2022. Том 13. № 9. 413–468. Подписной индекс по Объединенному каталогу "Пресса России" – 22765