Министерство науки и высшего образования Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Радиофизический факультет
Кафедра информационных технологий в исследовании дискретных структур

ДОПУСТИТЬ К ПРЕДСТАВЛЕНИЮ ГЭК
Руководитель ООП
д-р физ.-мат.наук, профессор
_____ С.П. Моисеева
«13» июня 2020 г.

НАУЧНЫЙ ДОКЛАД
об основных результатах подготовленной научно – квалификационной работы (диссертации)

ИСПОЛЬЗОВАНИЕ ЛОГИЧЕСКИХ СХЕМ В ЗАДАЧАХ АНАЛИЗА И СИНТЕЗА КОМПОНЕНТОВ ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ

по основной образовательной программе подготовки научно-педагогических кадров в аспирантуре
направление подготовки 09.06.01 – Информатика и вычислительная техника

Лапутенко Андрей Владимирович

Научный руководитель
д-р.тех.наук, профессор
_____ Н.В. Евтушенко
«13» июня 2020г.

Автор работы
аспирант
_____ А.В. Лапутенко

Томск – 2020

Диссертационная работа выполнена в федеральном государственном автономном образовательном учреждении высшего образования «Национальный исследовательский Томский государственный университет», на кафедре информационных технологий в исследовании дискретных структур

Научный руководитель: доктор технических наук, профессор,

Евтушенко Нина Владимировна

Общая характеристика работы

Актуальность темы

Исторически логические схемы являются одной из первых формальных моделей для анализа и синтеза цифровых систем и устройств, и исследование логических схем активно проводилось с начала 40х годов XX века [27, 37, 41, 45]. В последнее время по ряду причин интерес к исследованиям логических схем получил «новый виток». Одной из таких причин является возможность эффективного использования логических схем для анализа и синтеза программных и аппаратных компонентов киберфизических и телекоммуникационных систем, в частности, логические схемы можно использовать в качестве формальных моделей для программных и аппаратных компонентов с большим числом состояний при построении тестов с гарантированной полнотой. Еще одним направлением, получившим широкое распространение в последнее время, является использование логических схем для представления различных моделей машинного обучения [6], в частности, для оценки качества телекоммуникационных сервисов и определения характеристик доверия к приложениям, выполняющимся на устройствах с ограниченными вычислительными возможностями. Данная задача появилась в связи с высоким темпом развития технологии «интернета вещей» (Internet of Things, IoT) [7, 10], сутью которой является объединение большого числа малогабаритных устройств, генерирующих и обменивающихся информацией, которая достаточно часто является конфиденциальной, друг с другом (различные датчики, «умные» устройства и т.д.). Соответственно при решении задач безопасности важным шагом является определение характеристик доверия (англ. trust) к приложению, выполняющемуся на подобном устройстве с применением моделей машинного обучения. Поскольку подобные устройства снабжаются маломощными процессорами, небольшим количеством памяти и аккумуляторами небольшой емкости, это сильно затрудняет запуск на них дополнительных приложений, особенно таких ресурсоемких как программные реализации алгоритмов и моделей машинного обучения. В настоящее время для эффективного представления алгоритмов машинного обучения, в частности, нейронных сетей, предлагается использовать логические схемы [2, 33], точность предсказания которых оказывается близкой, а в некоторых случаях равной нейронной сети, по которой строится логическая схема.

Целью работы является разработка математических, программных и аппаратных средств анализа и синтеза компонентов телекоммуникационных систем на основе логических схем.

Методы исследования. Для решения задач анализа и синтеза компонентов телекоммуникационных систем используется аппарат дискретной математики, в частности, аппарат классической теории автоматов и математической логики. Оценка качества предложенных методов проводится с помощью компьютерных экспериментов.

Основные результаты исследования и положения, выносимые на защиту:

В диссертации решаются две задачи анализа и синтеза компонентов телекоммуникационных систем с использованием логических схем.

1. Метод синтеза тестов на основе логических схем, доставляющий тесты с гарантированной полнотой относительно неисправностей (временного) конечного автомата.
2. Метод синтеза логической схемы по обученной модели машинного обучения для оценки характеристик доверия к приложениям, выполняющимся на устройствах с ограниченными вычислительными возможностями; логическая схема далее может быть использована для аппаратной реализации модели машинного обучения.
3. Экспериментальные результаты по оценке эффективности предложенных методов.

Личный вклад в положения, выносимые на защиту.

Результаты получены диссертантом лично. В работах по теме диссертации диссертантом предложены ключевые научные идеи, реализованы и проведены эксперименты, в опубликованных статьях написаны части, в которых изложены выносимые на защиту результаты.

Научная новизна.

1. С использованием логических схем предложен метод синтеза тестов для дискретных систем на различных уровнях абстракции. Экспериментально исследовано качество тестов, построенных по логической схеме и модели (временного) конечного автомата.
2. Исследована эффективность построения мутационных тестов по модели логической схемы для проверки функциональных свойств дискретных систем, в том числе для верификации ПЛИС, реализованных с использованием FPGA технологии.
3. Предложен метод для представления моделей машинного обучения в виде логических схем для последующей эффективной аппаратной реализации по технологии FPGA.

Теоретическая и практическая ценность.

В работе предложены метод и алгоритмы для проверки функциональных требований к системам, поведение которых задается (временными) конечными автоматами на основе синтеза мутационных тестов по модели логической схемы. Метод опробован на тестировании микроконтроллерной реализации генератора импульсов запуска CCD камеры. Предложен метод для представления моделей машинного обучения в виде логических схем для последующей эффективной аппаратной реализации по технологии FPGA; аппаратная реализация может быть использована в устройствах с ограниченными вычислительными возможностями для оценки характеристик доверия к приложениям, которые выполняются на устройстве, в том числе, для компонентов «интернета вещей». Эксперименты, проведенные с программной реализацией IoT системы удаленного мониторинга температуры, показали эффективность предложенного подхода.

Апробация работы.

Основные положения и результаты были представлены на следующих конференциях и семинарах: Российская конференция с международным участием «Новые информационные технологии в исследовании сложных структур» (г. Екатеринбург, Россия, 2016; пос. Катунь, Россия, 2018), Международная конференция молодых специалистов по микро/нанотехнологиям и электронным приборам EDM (Алтайский край, Россия, 2017, 2018, 2019), семинар «Методы анализа и синтеза дискретных систем на основе логических схем» ИСП РАН (Москва, Россия, 2018), Международная конференция по программным технологиям ICSOFT (Порту, Португалия, 2018).

Содержание работы.

Диссертация состоит из введения, 4 глав, заключения и списка использованной литературы.

Во **введении** приводится общая характеристика работы, обосновывается актуальность исследований, формулируются цель и основные задачи диссертации, кратко излагаются полученные результаты, научная новизна работы, практическая ценность и выносимые на защиту положения.

В **первой главе** вводятся основные определения и обозначения.

Конечные системы переходов, такие, например, как конечные автоматы, активно используются при решении задач анализа и синтеза для различных систем, которые переходят из состояния в состояние под действием входных воздействий, производя при этом выходные реакции.

Формально под *конечным автоматом* или просто *автоматом* будем понимать пятерку [12, 30] $\mathcal{S} = (S, I, O, T_S)$, где S – непустое конечное множество *состояний*, I – непустое конечное множество входных символов, называемое *входным алфавитом*, O – непустое конечное множество выходных символов, называемое *выходным алфавитом*, $T_S \in S \times I \times O \times S$ – множество *переходов*. Четверка $(s, i, o, s') \in T_S$ называется *переходом* в автомате S и описывает ситуацию, когда на автомат, находящийся в состоянии s, поступает входной символ i под действием которого автомат переходит в состояние s' с выдачей выходного символа o. Автомат $\mathcal{S}$ с выделенным начальным состоянием s_0 называется *инициальным* и обычно описывает поведение систем, в которых существует надежный специальный сигнал СБРОС (*reset*), по которому система гарантированно переходит из любого состояния в начальное состояние.

Автомат называется *детерминированным*, если из каждого состояния s существует только один переход для каждого входного воздействия, определенного в данном состоянии. Детерминированные автоматы активно исследовались в 20-м веке, в частности, в области построения тестов с гарантированной полнотой для телекоммуникационных протоколов [1, 5, 4, 35], при разработке автоматизированных и автоматических методов синтеза управляющих систем, трансформирующих последовательности в одном (входном) алфавите в последовательности в другом (выходном) алфавите, и в этой области получен ряд основополагающих результатов [42, 44, 41]. В большинстве работ рассматриваются полностью определенные автоматы, когда для каждой пары $(s, i) \in S \times I$ существует пара $(o, s') \in O \times S$, такая, что $(s, i, o, s') \in T_S$; иначе автомат называется *частичным*. Чтобы избежать использования частичных автоматов, в ряде случаев частичный автомат различными способами, соответствующими рассматриваемой задаче, доопределялся до полностью определенного, и в нашей работе мы в большинстве случаев рассматриваем компоненты, поведение которых определено на каждой входной последовательности.

Под *временным автоматом* $\mathcal{S}$ понимается детерминированный автомат, в который добавлены входные и выходные таймауты. Таким образом, временной автомат есть семёрка $(S, I, O, \lambda_S, s_0, \Delta_S, \sigma_S)$, где S, I, и O – конечные непустые множества состояний, входных и выходных символов соответственно, s_0 – начальное состояние, $\lambda_S \subseteq S \times I \times O \times S \times Z$ – отношение переходов, Δ_S: $S \to S \times Z$ – функция таймаутов, и σ_S: $S \times I \times O \times S \to Z$ – функция выходных задержек (выходных таймаутов). Для λ_S через Z обозначено множество входных задержек (входных таймаутов), где Z есть множество натуральных чисел в объединении с числом 0. Если время ожидания в некотором состоянии превышает (входной) таймаут для этого состояния, то автомат может спонтанно перейти в другое состояние. В данной работе рассматриваются временные автоматы, у которых в каждом состоянии задержка выхода каждого перехода меньше величины таймаута.

Временным входным символом называется пара (i, t), где i – символ входного алфавита, t – (вещественное) время поступления входного символа после перехода автомата в текущее состояние. *Временным выходным символом* называется пара (o, d), где o – символ выходного алфавита, d – выходная задержка. Последовательность временных входных символов (i_1, t_1), (i_2, t_2) …, (i_n, t_n) называется *временной входной последовательностью*; последовательность временных выходных символов (o_1, d_1), (o_2, d_2) …, (o_n, d_n) называется *временной выходной последовательностью*. Аналогично конечным автоматам, временная выходная последовательность, соответствующая временной входной последовательности α, поступившей на автомат в состоянии s, называется (*выходной*) *реакцией* автомата в состоянии s на последовательность α.

В ряде случаев поведение временного автомата можно описать специальным автоматом [38], который называется *конечно автоматной абстракцией автомата с таймаутами* (*временного автомата*). В этом случае вводится специальный входной и выходной символ `I`, который соответствует единице времени, и все переходы в исходном временном автомате соответственно «растягиваются».

Классический конечный автомат для временного автомата строится в два этапа. На первом шаге строится временной автомат с таймаутами, но без задержек выходов: для каждого выходного символа o_i, выдаваемого с некоторой задержкой θ, в выходной алфавит вводится новый выходной символ $<o_i, \theta>$. На втором шаге выполняется построение классического конечного автомата по заданному временному автомату с таймаутами: каждый переход $s \xrightarrow{T} s'$ заменяется цепочкой переходов $s \xrightarrow{I/I} s_1 \xrightarrow{I/I} s_2 \xrightarrow{I/I} \dots \xrightarrow{I/I} s_{T-1} \xrightarrow{I/I} s'$, где каждое промежуточное состояние является новым состоянием и копией состояния *s* по всем входо-выходным парам, кроме `I/I`.

Конечный автомат, построенный по описанным выше правилам, является конечным автоматом специального вида и имеет ряд особенностей: во-первых, мы выделяем в этом автомате состояния, в которых автомат не может находиться дольше одного такта времени; во-вторых, входной и выходной алфавиты пересекаются, так как содержат один и тот же выделенный символ `I`. Этот символ обозначает отсутствие подачи какого либо входного символа на автомат в течение 1 такта времени, а также, отсутствие какой либо реакции автомата. Последнее не вызывает никаких проблем, т.к. эти символы можно снабдить различными индексами для обозначения входного и выходного символов.

Одним из недостатков модели конечного автомата при решении задач анализа и синтеза для компонентов киберфизических и телекоммуникационных систем является его размерность. Современные программы достаточно быстро решают задачи синтеза и анализа автоматов, размеры которых не превосходят 1000 состояний и примерно такого же числа входных и выходных символов. Для автоматов бóльших размерностей активно используется представление автомата в виде логической схемы [2, 33]. Логическая схема состоит из логических элементов, где под (логическим) *элементом* понимается (*m*, 1)-полюсник, который имеет *m* входных полюсов (входов), один выходной полюс (выход) и реализует некоторую логическую функцию. В данной работе элемент может быть комбинационным (*вентилем*), т.е. реализовывать некоторую элементарную булеву функцию (AND/OR/NOT/XOR/XNOR/BUFF/NAND/NOR), либо элемент является элементом задержки, который имеет два состояния 0 и 1, два входных и два выходных символа. Такой элемент задержки часто называется *D-триггером* (обозначение: *D* или *d*). Следующим состоянием элемента задержки (триггера) является значение текущего входного символа, выходным символом является значение текущего состояния. Под *логической схемой* понимается пара $<E, R>$, где E – совокупность элементов, имена которых попарно различны, и R – отношение эквивалентности, заданное на множестве полюсов элементов из E. Блоки разбиения R, содержащие выходной полюс, также могут быть объявлены выходными полюсами сети. В данной работе рассматриваются схемы с синхронным функционированием, т.е. предполагается исследование специального класса логических сетей, в которых следующее состояние и выходной вектор вычисляются за один такт. *Комбинационная схема* состоит из взаимосвязанных логических элементов, не имеет элементов памяти и обратных связей, и реализует в общем случае систему булевых функций. *Последовательностные схемы* (схемы с памятью) включают как комбинационную часть (комбинационную схему), так и элементы задержки (триггеры). Последовательностная схема реализует множество последовательностных функций, то есть выходная реакция схемы зависит не только от текущего входного вектора, но и от предыдущих входных последовательностей. Пример последовательностной схемы изображен на рисунке 1. Данная логическая схема имеет один вход, два элемента задержки и один выход.

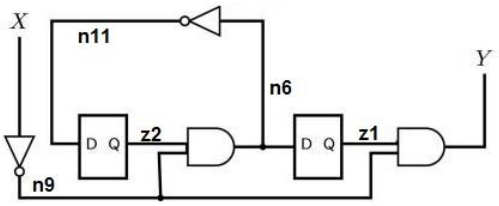

Рисунок 1. Пример последовательностной схемы

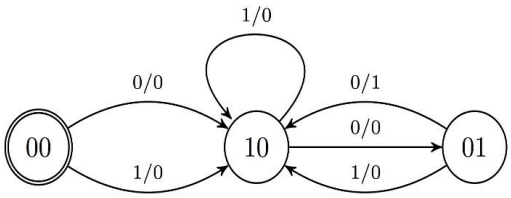

Рисунок 2. Конечный автомат, описывающий поведение схемы на рисунке 1

Конечный автомат, соответствующий логической схеме, можно построить путем моделирования выходных реакций схемы во всех состояниях в ответ на входные воздействия. Множество состояний конечного автомата состоит из булевых векторов (при необходимости хешированных целыми числами), представляющих комбинации значений элементов задержки, достижимые из начальной комбинации (начального состояния). Соответственно, множество входных символов автомата содержит булевы векторы, которые могут быть поданы на входы схемы. Например, конечный автомат, описывающий поведение логической схемы, изображенной на рисунке 1, представлен на рисунке 2. Поведение цифровой схемы можно описать с использованием HDL (Hardware Description Language) языков, таких как языки Verilog и VHDL и форматов описания цифровых схем, таких как BLIF (Berkeley Logic Interchange Format) формат и bench. Для построения автомата по логической схеме можно использовать программные инструменты, которые принимают в качестве входных данных логическую схему в bench формате (ISCAS'98), и возвращают автомат в KISS формате, где состояния абстрактные, а входные и выходные символы закодированы булевыми векторами. Для описания детерминированного автомата логической схемой состояния, входные и выходные символы кодируются булевыми векторами подходящей длины и соответственно вычисляются функции переходов и выходов автомата в виде системы булевых функций. В частности, использование представления логических схем в виде BDD уже дало возможность рассматривать дискретные системы с тысячами состояний, входных и выходных символов, однако число элементов задержки обычно ограничивалось требованиями решаемой задачи, входных и выходных портов было не более 50. Появление методов и алгоритмов решения задач анализа и синтеза для логических схем на основе решения проблемы выполнимости (SAT проблемы) [23] предоставило возможность рассматривать схемы с большим числом элементов задержки, входных и выходных портов.

Во **второй** главе исследуется задача синтеза тестов для компонентов киберфизических и телекоммуникационных систем на высоком и низком уровнях абстракции, а именно на уровне логических схем и конечных автоматов. Построенные тесты используются для проверки функциональных требований, в том числе компонентов, реализованных с использованием технологии FPGA [36]. Преимуществом модели конечного автомата является тот факт, что тесты строятся по ней без явного перечисления неисправностей, только по автомату-спецификации. Недостатком является размерность модели, не позволяющая строить тесты для компонентов с числом состояний больше 1000. Достоинством модели логической схемы является ее масштабируемость. Например, для системы с 1000 состояний достаточно использовать логическую схему с числом элементов задержки не больше 10. Результаты, представленные в данной главе, опубликованы в работах [39, 18, 19, 31].

В связи с развитием технологии FPGA множество неисправностей, рассматриваемых в логических схемах, было значительно расширено, поскольку константные неисправности [41] были, в первую очередь, адекватными только для релейно-контактных схем. Тесты, строящиеся по моделям высокого уровня, таким как, например, расширенные автоматы, обладают низкой полнотой относительно ошибок в реализациях компонентов цифровых устройств [28]. Согласно исследованиям, такие тесты

оставляют необнаруженными большое число функциональных ошибок в программном обеспечении [24]. Проверка качества тестов, построенных конечно автоматными методами, относительно неисправностей в логических схемах проводилась в основном относительно (одиночных) константных неисправностей [3, 13, 25, 26]. Нам не известны подобные результаты относительно неисправностей «перемычек» и трудно обнаружимых неисправностей. Поэтому, в настоящей работе мы исследуем качество тестов, построенных на различных уровнях абстракции, в том числе на уровне классического автомата и логической схемы. Мы рассматриваем мутационный подход к синтезу тестов по модели логической схемы, при котором строятся различающие последовательности для эталонной логической схемы и неэквивалентных ей схем, содержащих специально внедренные мутации одного из известных типов. Такой подход позволяет единым образом строить тесты с гарантированной полнотой относительно любых мутаций в логической схеме. Еще одним аргументом для реализации новых методов построения тестов для систем, поведение которых описано конечными системами переходов, на основе логических схем является возможность их использования для неклассических автоматных моделей, таких как расширенные и временные автоматы, для которых методы синтеза проверяющих тестов с гарантированной полнотой развиты недостаточно хорошо, в то время как конечно автоматный уровень достаточно хорошо исследован при построении таких тестов.

В диссертации исследованы свойства тестовых последовательностей, построенных для верификации логических компонентов, используемых в киберфизических и телекоммуникационных системах относительно наиболее распространенных моделей ошибок трех типов, а именно, одиночных константных неисправностей, одиночных неисправностей «перемычек» и одиночных неисправностей при замене одного вентиля схемы (а именно, одного единичного набора из таблицы истинности подходящей булевой функции), т.н. трудно обнаружимые неисправности. Именно неисправности при замене одного вентиля схемы стали популярны с развитием технологии FPGA, в которой описание схем дается на языках высокого уровня (Verilog, VHDL), и разработчик может легко допустить ошибку при описании одного логического элемента.

Мутационный подход к синтезу тестов для логических схем подразумевает, что для каждой неисправности генерируется мутант спецификации, для которого на следующем шаге строится различающая последовательность. В работе сравниваются свойства тестов, построенных на основе кратчайших различающих последовательностей для обнаружения мутаций в описании поведения логических схем и таких последовательностей, сгенерированных псевдослучайным образом с использованием системы ABC. Эксперименты проводились для набора контрольных примеров (бенчмарков) b01-b10, которые представляют собой компоненты киберфизических систем, спроектированных для различных приложений; эти компоненты включают схемы, направленные на обработку данных, полученных от сенсорных датчиков, системы распределения нагрузки и др.

В ходе экспериментов [18] сравнивались три подхода к построению так называемых полных тестов, т.е. тестов, обнаруживающих все возможные мутации заданного класса для эталонного описания логической схемы. В первом случае для каждой мутации находится обнаруживающая ее кратчайшая тестовая последовательность, во втором случае – тестовая последовательность генерируется псевдослучайным образом и, как правило, является более длинной, в третьем случае – первая часть последовательностей определенной длины генерируется случайным образом, и для каждой мутации, не обнаружимой данной последовательностью, достраиваются различающие последовательности. Наиболее эффективным является третий подход. Как показывают проведенные эксперименты, длина полного теста, построенного с использованием комбинированного алгоритма, оказывается минимальной.

Предполагая, что внесение неисправности в логическую схему, отражается на ее функционировании, тесты, построенные на уровне конечного автомата, соответствующего логической схеме, должны обнаруживать такую неисправность. По результатам экспериментов [19, 39] полнота обнаружения различных видов неисправностей в логической схеме тестом, построенным на основе обхода графа переходов соответствующего ей конечного автомата, составляет более 90% для схем из пакета бенчмарков ITC'99 и логическим схем, построенным по случайно сгенерированным автоматам с числом состояний от 5 до 20 и числом входов от 2 до 16. Таким образом, тесты, построенные на высоком уровне абстракции (по модели конечного автомата), имеют достаточно высокий процент покрытия относительно трех типов неисправностей в логических схемах, но, тем не менее, не достигают 100%.

В связи с этим, в работе предлагается метод для построения тестов для последовательностных схем, которые бы являлись полными относительно неисправностей трех рассматриваемых типов логической схемы, а также относительно выходных ошибок в конечном автомате, соответствующем данной логической схеме. Если конечный автомат, построенный по логической схеме, оказывается слишком большим, то для построенного теста устанавливается, относительно какой автоматной модели неисправности тест является полным.

Пусть поведение логической схемы описывается автоматом $\boldsymbol{S}$, в котором множество состояний S есть подмножество всех двоичных наборов длины n, где n – число элементов задержки, и входной алфавит I содержит все двоичные наборы длины p, где p – число внешних входов логической схемы. Для каждой неисправности логической схемы построим соответствующий мутант; различающая последовательность для эталонной логической схемы и мутанта (если существует) строится с использованием одной из систем логического синтеза и верификации. Построенные последовательности собираются в единый тест TS, который можно дополнительно оптимизировать. Поведение логической схемы моделируется на последовательностях теста TS и строится соответствующий конечный автомат $M_{LC}(TS)$. В общем случае автомат $M_{LC}(TS)$ детерминированный и частично определенный. Построим разбиение π множества $S \times I$ на классы таким образом, что каждый класс множества содержит некоторую пару «состояние, входной символ», для которой определено поведение автомата $M_{LC}(TS)$. Рассмотрим модель неисправности $<\boldsymbol{S}, \cong, FD_\pi>$, где $\boldsymbol{S} = (S, I, O, next\_state_s, out_s, s_0)$ – полностью определенный автомат-спецификация, $\cong$ – отношение эквивалентности. Область неисправности FD_π содержит каждый полностью определенный автомат $\boldsymbol{P}$ с входным алфавитом I, множество P состояний которого есть подмножество множества S и в котором возможны только выходные ошибки относительно эталонного автомата $\boldsymbol{S}$, причем для каждого класса b из π справедливо: либо выходной символ в автомате $\boldsymbol{P}$ «правильный» для всех пар класса, либо для каждой пары класса в автомате $\boldsymbol{P}$ есть ошибка в выходном символе.

Теорема 2.1. Проверяющий тест TS обнаруживает каждый автомат из класса FD_π, не эквивалентный эталонному автомату $\boldsymbol{S}$, т.е. является полным проверяющим тестом относительно модели неисправности $<\boldsymbol{S}, \cong, FD_\pi>$.

Следствие. Если автомат $M_{LC}(TS)$ является полностью определенным, то тест TS является полным относительно всех выходных неисправностей в логической схеме.

Если автомат, соответствующий логической схеме, не удается построить ввиду его громоздкости, то можно, либо довольствоваться полнотой теста согласно теореме 2.1, или достраивать тест до покрытия тех переходов, которые достаточно просто промоделировать. При этом согласно проведенным экспериментам, после увеличения длины теста в 2-4 раза можно полагать, что большое количество переходов в автомате окажется покрытым.

В работе приводятся экспериментальные результаты по оценке времени генерации теста конечно автоматным методом и мутационным способом для схем, показывающие

намного бóльшую масштабируемость конечно автоматного способа при использовании обхода графа переходов и сохраняющие полноту построенных тестов порядка 90% относительно неисправностей в соответствующей логической схеме. Для построения полных тестов можно использовать тест, построенный по W-методу для увеличения числа состояний на 20%. Однако для автоматов с числом состояний более 1000 и числом входных символов более 4 при использовании W-метода даже в предположении увеличения числа состояний на единицу тесты строятся несколько часов. Поэтому, если логическая схема имеет более 10 элементов задержки, то можно использовать подход, интегрирующий методы синтеза тестов по модели конечного автомата и методы синтеза теста по логической схеме для построения качественного теста: тест, построенный на первом шаге обходом графа переходов автомата, соответствующего логической схеме, на следующем шаге дополняется последовательностями, обнаруживающими мутации логической схемы, не обнаруженные на первом шаге.

Таким образом, результаты данной главы подтверждают равноправность использования обоих уровней абстракции при построении проверяющих тестов: конечных автоматов и логических схем, и предлагают способ интеграции тестов, построенных на различных уровнях абстракции, с учетом достоинств и недостатков синтеза тестов на каждом из уровней.

В **третьей** главе рассматривается метод синтеза тестов для еще одного уровня абстракции, а именно, для уровня временного автомата, который является расширением модели классического автомата. Результаты, представленные в данной главе, опубликованы в работах [32, 34, 8, 15, 16].

Поскольку большинство систем строятся как композиции более простых в некотором смысле подсистем, на первом этапе исследуется проблема построения параллельной композиции для автоматов с таймаутами. Операция параллельной композиции достаточно часто используется при описании поведения взаимодействующих программных систем. В этом случае автоматы-компоненты работают в режиме диалога, при наличии медленной внешней среды, т.е. когда входное воздействие на систему подается только после получения реакции системы на предыдущее входное воздействие. При решении задач анализа, в том числе для построения проверяющих тестов для системы взаимодействующих детерминированных конечных автоматов с использованием классических методов, желательно, чтобы поведение такой композиции описывалось также детерминированным автоматом.

Известно, что, если компоненты композиции являются детерминированными полностью определенными автоматами, то поведение параллельной композиции описывается полностью определенным детерминированным автоматом при наличии «медленной внешней среды» и отсутствии осцилляций, т.е. бесконечного диалога между компонентами. В этом случае состояния автомата-композиции обычно представляют собой пару состояний компонент. Задача построения композиций автоматов с таймаутами рассматривалась в ряде работ [38], где, в частности, предлагается перейти от временных автоматов к соответствующим полуавтоматам, представляющим регулярные языки, и построить композицию путём пересечения полуавтоматов. Однако нами показано, что данный подход становится неадекватным в том случае, когда временная переменная может принимать не только целочисленные значения.

Теорема 3.1. Множество детерминированных конечных автоматов с таймаутами не является замкнутым относительно операции параллельной композиции.

Причиной утверждения теоремы 3.1 является тот факт, что компонента, не взаимодействующая с внешней средой «накапливает» вещественное значение временной переменной и в зависимости от момента поступления внешнего входного воздействия может «перейти» или «не перейти» целочисленный порог, изменяющий поведение компоненты. Вообще говоря, последнее согласуется с результатами в области временных полуавтоматов, где показано, что не для всякого временного полуавтомата с несколькими

временными переменными существует эквивалентный полуавтомат с одной временной переменной.

Тем не менее, описание композиции автоматов с таймаутами в виде детерминированного автомата возможно в случае, когда временная переменная принимает только целочисленные значения, а также для определенных классов автоматов.

Теорема 3.2. Параллельная композиция автоматов с таймаутами S и P может быть описана детерминированным временным автоматом, если в компоненте S не существует перехода $(s, i, s', o) \in h_S$ (в компоненте P не существует перехода $(p, i, p', o) \in h_P$), где i и o – внешние входной и выходной символы, и в построенной параллельной композиции в каждом состоянии входной таймаут больше, чем выходные задержки для обработки каждого входного символа.

Если поведение параллельной композиции автоматов с таймаутами описывается детерминированным автоматом, то тесты для такой композиции можно строить на основе логической схемы, построенной по конечно автоматной абстракции. В этом случае используется мутационный подход к синтезу тестов для логических схем, описанный в главе 2 настоящей работы, где множество мутаций логической схемы достаточно широкое.

Вышеописанный подход был использован при синтезе тестов для микроконтроллерной системы, поведение которой описывается автоматом с таймаутами. Система представляет собой реализацию коммутирующего генератора на микроконтроллере ATmega8 и используется для управления CCD-камерой при исследовании колебаний биморфного пьезоэлемента в виде пластины. Электрический сигнал синусоидальной формы подается на пластину, заставляя ее колебаться. Тот же синусоидальный сигнал, пропущенный через триггер Шмитта, подается на микроконтроллер в виде прямоугольных импульсов со скважностью равной 2. Для того чтобы получить изображение положения пластины, микроконтроллер генерирует на выходе синхроимпульсы для запуска CCD-камеры в соответствии с сигналами на входе. Выходной сигнал генерируется относительно фронтов и спадов входного сигнала (либо только по фронтам, либо только по спадам, либо по фронтам и спадам). Главное ограничение системы – частота сигнала на выходе не должна превышать 50 Гц.

Для спецификации системы управления фотокамерой была выбрана модель конечного автомата с таймаутами. Таймауты помогают описать состояния системы, в которых происходит превышение периода входных импульсов, соответствующего частоте 50 Гц. Входной алфавит данной системы представлен множеством {*RE, FE*}, где *RE* – фронт входного сигнала, *FE* – спад входного сигнала, выходной алфавит представлен множеством {*P, nul*}, где *P* – импульс на выходе микроконтроллера, *nul* – отсутствие выходного действия. В качестве единицы времени выбран временной промежуток в 1 мс. Значение таймаута T1 – 20 единиц.

По временному автомату с таймаутами, описывающему систему управления CCD камерой, была построена конечно автоматная абстракция, как описано в главе 1, которая имеет 105 состояний. По конечно автоматной абстракции временного автомата с таймаутами была построена логическая схема с двумя входами, одним выходом, содержащая 635 двухвходовых логических элементов и 7 элементов задержки. С помощью мутационного подхода был построен полный тест относительно трех типов неисправностей в логических схемах. Построенный тест содержал 3201 входных символов.

Для оценки полноты построенного теста относительно распространенных ошибок в исходных кодах программ для микроконтроллеров были проведены следующие эксперименты. Рассматривались следующие виды ошибок в программах: 1) неверная установка предделителя частоты тактирования таймера; 2) ошибки в назначении портов ввода/вывода микроконтроллера; 3) ошибки в операторах исходного кода для микроконтроллера. В данном случае использовались следующие операторы мутирования

исходного кода программ: 1) замена арифметических операторов (« + » заменяется на « - » и наоборот, « * » заменяется на « / » и наоборот); 2) замена операторов сравнения (каждый из операторов « > », « < », « == », « != » заменяется по очереди на каждый из остальных).

Для системы рассматривались только мутанты первого порядка. Всего было сгенерировано 54 мутанта. С помощью построенных тестовых последовательностей все внесенные неисправности удалось обнаружить, что говорит о хорошем качестве тестов, построенных по логической схеме временного автомата, позволяющих обнаружить распространенные ошибки в исходных кодах программ для микроконтроллеров.

В **четвертой** главе исследуются вопросы использования логических схем применительно к оценке характеристик доверия (англ. trust) к приложениям, выполняющимся на устройствах с ограниченными вычислительными возможностями в «интернете вещей». Результаты, представленные в главе, опубликованы в работах [40, 22].

В настоящее время, в связи с интенсивным развитием в сфере информационных технологий такого направления, как интернет вещей (Internet of Things, IoT) [7, 10], увеличивается использование устройств с ограниченными вычислительными возможностями. Реализация на данных устройствах методов классификации, основанных на технологиях машинного обучения, вызывает большой интерес, так как данные устройства активно взаимодействуют друг с другом и часто возлагают на подобные себе узлы сети выполнение критических операций. В настоящей главе исследуются вопросы использования логических схем применительно к возможному прибавлению к устройствам с ограниченными вычислительными возможностями в «интернете вещей» специальной «добавки», оценивающей характеристики доверия (англ. trust) к приложениям, выполняющимся на устройстве.

В литературе описаны различные методики машинного обучения, применяемые для решения различных задач классификации, например, системы опорных векторов (Support vector machine, SVM). Различными исследователями предложены алгоритмы для обучения систем опорных векторов, нейронных сетей, и др. с высокой точностью предсказания независимо от исходных данных. Главным недостатком подобных моделей в случае их применения в устройствах с ограниченными вычислительными возможностями является их высокая требовательность к вычислительным ресурсам устройства. Поэтому, в данной работе предложено использовать логические схемы, которые строятся на основе обученной с достаточной точностью модели машинного обучения. Поскольку число параметров, которые существенно влияют на оценку характеристик доверия, в устройствах с ограниченными вычислительными возможностями, как правило, не слишком большое, использование логических схем оказалось эффективным.

При реализации модели машинного обучения необходимо построить логическую схему по множеству значений параметров и сопоставленных им значений меток. Первый шаг заключается в кодировании каждого набора значений параметров булевым вектором. Каждому такому булеву вектору ставится в соответствие булев вектор, кодирующий значение метки для соответствующего набора значений параметров. После обучения модели на всех значениях параметров или в заданной области, полученные булевы вектора формируют таблицу истинности для системы булевых функций, по которой и нужно синтезировать логическую схему любым из известных методов. В данной работе для синтеза логической схемы используется система логического синтеза и верификации ABC.

Рассмотрим пример на рисунке 3, на котором представлен набор данных D для двух параметров $x1$ и $x2$, для значений которых нужно провести классификацию на два класса $o \in \{0, 1\}$, обозначенных как **x** и **o**. Заданные значения параметров были нормализованы добавлением числа 2 ко всем значениям.

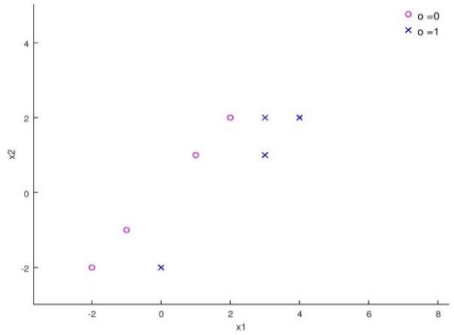

Рисунок 3 Пример набора данных *D*

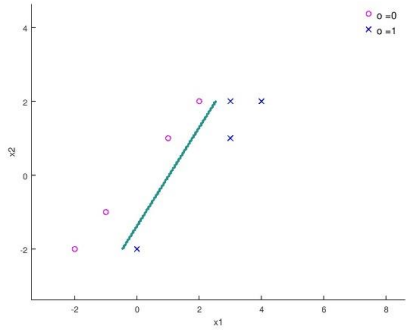

Рисунок 4 Линейная граница решения, полученная с помощью SVM

Система опорных векторов, обученная по одному из известных алгоритмов на данных из множества *D* доставила линейную границу решений (рисунок 4). По методу, представленному выше была синтезирована логическая схема, имеющая точность классификации аналогичную системе опорных векторов для значений параметра *x*1 из интервала [-2, 4], для *x*2 из интервала [-2, 2]. Построенная логическая схема содержит 6 входов, один выход и 16 двухвходовых элементов. В общем случае, схема *C*, синтезированная по наборам, полученным после обучения модели *M*, имеет точность предсказаний аналогичную модели *M* в пределах границ данных, по которым она была синтезирована. Такая комбинационная логическая схема может быть реализована аппаратно, и синтезированное устройство может быть интегрировано в устройство с ограниченными вычислительными возможностями. Необходимо отметить, что предлагаемый алгоритм может использоваться, вообще говоря, для работы с любой предсказывающей моделью с разумно ограниченным обучающим множеством. В работе также рассматривается композиция логических схем для случая, когда имеется несколько логических схем, построенных по моделям машинного обучения, обученных для классификации данных из пересекающихся множеств. Такая композиция помогает повысить точность классификации значений по сравнению с исходными моделями.

Для того чтобы показать, насколько быстрыми и масштабируемыми являются логические модели по сравнению со сложными самообучающимися моделями такими, как искусственные нейронные сети (ANN) и системы опорных векторов (SVM), была проведена серия экспериментов. Результаты экспериментов показывают, что самообучающиеся модели при вычислении определенных характеристик доверия проигрывают в производительности логической схеме, которая симулируется программно.

Для симуляции логической схемы была использована программа ABC 1.01 (команда *sim*). Моделирование схемы на входных наборах производилось 10000 раз для измерения среднего времени работы, которое составило 0.02 с. Таким образом, в среднем на обработку одного входного набора уходит $2 \cdot 10^{-6}$ с. При использовании среды GNU Octave 3.4.3 с пакетом LibSVM 3.22, время, затрачиваемое на симуляцию SVM на том же самом входном наборе, составило $9{,}5 \cdot 10^{-4}$ с. Таким образом, производительность логической схемы превосходит модель SVM в 424 раза.

Во второй серии экспериментов использовалась программная реализация нейронной сети на языке C и программная реализация логической схемы на языке C, получаемая по Verilog описанию схемы. В ходе эксперимента было сгенерировано несколько нейронных сетей с различным количеством нейронов в скрытом слое. Каждая нейронная сеть моделировалось на всех входных наборах и строилась соответствующая логическая схема, которая далее представлялась программой на языке *C*. В таблице 1 представлено среднее время, затрачиваемое нейронной сетью и соответствующей логической схемой на обработку одного входного набора.

Таблица 1 Время обработки одного входного набора нейронной сетью и логической схемой

Скрытые нейроны	Время работы ANN, мс	Время работы схемы, мс	Ускорение схемы
10	1,89	1,55	1,22
20	2,02	1,56	1,29
30	2,14	1,55	1,38
40	2,23	1,6	1,39
50	2,25	1,57	1,43
60	2,42	1,55	1,56
70	2,46	1,56	1,58
80	2,62	1,56	1,68
90	2,72	1,54	1,76
100	2,80	1,45	1,93

Как показывают проведенные эксперименты, во многих случаях даже оптимизированные реализации самообучающихся моделей значительно проигрывают оптимизированным программным реализациям логической схемы. Более того, учитывая подход, по которому синтезируется схема, время предсказания логической схемы не зависит от сложности модели, по которой синтезируется схема.

Значения переменных в исходном коде программы могут отражать показатели доверия к данному приложению, однако зависимость является достаточно сложной, и характеристики доверия достаточно часто оцениваются с использованием моделей машинного обучения. Соответственно, рассмотренный в предыдущем разделе подход можно использовать для оценивания характеристик доверия к приложению, выполняемому на некотором устройстве, в том числе в сети интернета вещей. Одним из наиболее важных аспектов, влияющих на характеристики доверия, является объем ресурсов, потребляемых данным приложением, и, вообще говоря, можно использовать различные параметры для оценки потребления ресурсов. Мы рассматриваем 5 таких параметров: размер кучи, использование ЦПУ (центральное процессорное устройство), использование диска, потребление энергии.

Были проведены эксперименты с системой удаленного мониторинга температуры по протоколу Ethernet, реализованной на микроконтроллере LPC4088. Запущенное на микроконтроллере приложение постоянно считывает значения датчика температуры и постоянно прослушивает HTTP порт на предмет входящих соединений, выполняя функционал HTTP-сервера. При появлении HTTP запроса, приложение отправляет HTTP-клиенту сообщение с текущим значением температуры.

Для рассматриваемого приложения были определены референсные значения потребляемых параметров. На основе референсных значений диапазон допустимых значений рассматриваемых параметров был разбит на поддиапазоны, соответствующие обычному режиму работы приложения и режимам с отклонениями от обычного. Для значений указанных выше параметров предполагалось определить уровень доверия, определяемый двумя значениями 0 (не доверяю) и 1 (доверяю). Множество всевозможных входных наборов значений параметров составляло 108 наборов, на каждом из которых значение характеристики доверия определено. На половине наборов из исходного множества была обучена нейронная сеть, и по нейронной сети была построена логическая схема. Значения, выдаваемые схемой на второй половине наборов, сравнивались со значениями в исходном множестве данных. Результаты эксперимента показали высокую точность модели логической схемы, построенной по нейронной сети, а также превосходство по скорости обработки входных данных.

Таким образом, глава 4 содержит предложенный метод реализации моделей машинного обучения в виде логической схемы, воспроизводящей поведение сложной самообучающейся модели. Такая «добавка» может использоваться как масштабируемое

устройство для вычислительно сложного предсказания/классификации при интеграции в устройство с ограниченными вычислительными возможностями, работающего в интернете вещей. Отметим, что описанный подход является общим и подходит для реализации любой модели машинного обучения на устройстве с ограниченными вычислительными возможностями.

В **заключении** приводятся основные результаты, полученные в диссертации и направления для дальнейших исследований.

На основе логических схем предложен метод синтеза тестов для проверки функциональных свойств дискретных систем, в том числе для верификации ПЛИС, реализованных с использованием FPGA технологии на различных уровнях абстракции; экспериментально исследовано качество тестов для дискретных систем построенных по логической схеме и модели (временного) конечного автомата. Экспериментально показано высокое качество таких тестов на примере тестирования микроконтроллерной системы управления CCD-камерой, поведение которой существенно зависит от временных аспектов.

Предложен метод реализации моделей машинного обучения в виде логических схем для последующей эффективной аппаратной реализации по технологии FPGA в том числе для устройств с ограниченными вычислительными возможностями при оценке характеристик доверия для компонентов «интернета вещей». Экспериментальные результаты показали превосходство по скорости работы (в некоторых случаях в сотни раз) реализации в виде логических схем таких моделей машинного обучения как система опорных векторов и нейронная сеть по сравнению с исходными программными реализациями данных моделей, в том числе на примере задачи определения характеристик доверия к приложению для удаленного мониторинга температуры через интернет, выполняющемуся на микроконтроллере с ограниченными вычислительными возможностями.

Для дальнейших исследований представляет интерес использование результатов главы 3 для построения тестов с гарантированной полнотой для расширенных автоматов, поскольку, расширенные автоматы являются наиболее близкой моделью по уровню абстракции к программным системам, однако разработка автоматических методов синтеза тестов с гарантированной полнотой для расширенных автоматов по-прежнему остаются актуальной задачей. Другой интересной задачей является реализация в виде логической схемы самообучающихся моделей без учителя, например, искусственных нейронных сетей, которые в настоящее время набирают популярность в использовании в различных областях обработки данных.

Публикации по теме работы

Статьи в журналах, включенных в Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени кандидата наук:

1. Лапутенко А. В., Винарский Е. М. Синтез тестов для цифровых систем на высоком и низком уровнях абстракции // Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика. 2019. № 47. С.110–117. DOI: 10.17223/19988605/47/13.
2. Laputenko, A.V., López, J.E., Yevtushenko, N.V. Verifying Digital Components of Physical Systems: Experimental Evaluation of Test Quality // Russian Physics Journal. 2017. Vol. 60, № 11. P. 2012-2018. DOI: 0.1007/s11182-018-1317-6.

3. M. L. Gromov, N. V. Yevtushenko, and A. V. Laputenko., Testing cyber-physical systems using timed finite state machines, Russian Physics Journal, Vol. 59, No. 12, April, 2017, pp. 2181-2182. Статья в журнале, проиндексированная в SCOPUS, Q3.
4. А.С. Твардовский, А.В. Лапутенко О возможностях автоматного описания параллельной композиции временных автоматов. Труды Института системного программирования РАН, том 30, вып. 1, 2018, стр. 25-40. DOI: 10.15514/ISPRAS-2018-30(1)-2.

В других изданиях:

1. Лапутенко А.В., Громов М.Л., Торгаев С.Н. Реализация и тестирование системы сигнализации на базе микроконтроллера STM32F407VG //Известия вузов. Физика. 2016. Т. 59, № 8/2. С. 61-64.
2. Громов М.Л., Евтушенко Н.В., Лапутенко А.В. Использование временных автоматов при тестировании киберфизических систем// Изв. вузов. Физика. – 2016. – Т. 59. – № 12. – С. 174-176.
3. Лапутенко А. В., Лопез Х. Е., Евтушенко Н. В. Обработка экспериментальных данных при верификации компонентов физических систем: оценка качества тестовых последовательностей. // Изв. вузов. Физика. – 2017. – Т. 60. – № 11. – С. 146-151.
4. López J., Laputenko A., Kushik N., Yevtushenko N. and Torgaev S. (2018). Scalable Supervised Machine Learning Apparatus for Computationally Constrained Devices. In Proceedings of the 13th International Conference on Software Technologies - Volume 1:ICSOFT, ISBN 978-989-758-320-9, pages 518-528. DOI: 10.5220/0006908905520562.
5. Jorge Lopez, Evgeny Vinarsky, Andrey Laputenko. On the Fault Coverage of High-level Test Derivation Methods for Digital Circuits //18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2017: proceedings Erlagol, 29 june - 3 july 2017. Novosibirsk: NSTU publisher, 2017. P. 184-189.
6. Andrey V. Laputenko, Timofey D. Petukhov, Nikolai A. Vasnev. Testing Microcontroller Based Physical Systems Using Finite Transition Models //19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2018: proceedings Erlagol, 29 june - 3 july 2018. Novosibirsk: NSTU publisher, 2018. P. 203-206, ISBN 978-1-5386-5020-2.
7. Evgenii Vinarskii, Andrey Laputenko, Jorge López, Natalia Kushik. Testing Digital Circuits: Studying the Increment of the Number of States and Estimating the Fault Coverage //19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2018: proceedings Erlagol, 29 june - 3 july 2018. Novosibirsk: NSTU publisher, 2018. P. 220-224, ISBN 978-1-5386-5020-2.
8. Andrey V. Laputenko. Logic Circuit Based Test Derivation for Microcontrollers //20th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2019: proceedings Erlagol, 29 june - 3 july 2019. Novosibirsk: NSTU publisher, 2019. P. 70-73, ISBN 978-1-7281-1753-9.

Список литературы

1. Bochmann G. V. Protocol testing: review of methods and relevance for software testing / G. V. Bochmann, A. Petrenko // Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis. – USA, 1994. – P. 109–124.

2. Brayton R. K., G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "Invited Paper: Multilevel logic synthesis," Proc. IEEE, vol. 78, no. 2, pp. 264-300, Feb. 1990.
3. Brzozowski J.A., H. Jurgensen. A model for sequential machine testing and diagnosis // Journal of Electronic Testing: Theory and Applications, No.2, 1992, pp. 219-234.
4. Cavalli A. Fault detection within a component of a system communicating FSMs / A. Cavalli, S. Prokopenko, N. Yevtushenko // Proceedings of the 14th International conference TestCom. – 2002. – P. 317–332.
5. Chow T. S. Testing software design modeled by finite-state machines // IEEE Transactions on Software Engineering. – 1978. – Vol. 4. – № 3. – P. 178–187.
6. Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. Foundations and Trends® in Signal Processing, 7(3–4):197–387.
7. Evans D, "The Internet of Everything: How More Relevant and Valuable Connections Will Change the World," Cisco Internet Business Solutions Group (IBSG), Cisco Systems, Inc., San Jose, CA, USA, White Paper 2012.
8. Gromov M. L., Yevtushenko N. V. and Laputenko A. V. Testing cyber-physical systems using timed finite state machines, Russian Physics Journal, Vol. 59, No. 12, April, 2017, pp. 2181-2182
9. Gromov, M. L., N. V. Yevtushenko, and A. V. Laputenko., Testing cyber-physical systems using timed finite state machines, Russian Physics Journal, Vol. 59, No. 12, April, 2017, pp. 2181-2182
10. Höller Jan et al., From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence, 1st ed. London, United Kingdom: Academic Press Ltd, 10 Apr 2014
11. Jorge Lopez, Evgeny Vinarsky, Andrey Laputenko. On the Fault Coverage of High-level Test Derivation Methods for Digital Circuits //18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2017: proceedings Erlagol, 29 june - 3 july 2017. Novosibirsk: NSTU publisher, 2017. P. 184-189.
12. Kam T., T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, Synthesis of Finite State Machines: Functional Optimization, Boston, MA: Kluwer Academic Publishers, 1997.
13. Kohavi I., Z. Kohavi. Detection of Multiple Faults in Combinational Logic Networks. IEEE Transactions on Computers, Vol. C-21, No.6, 1972.
14. Laputenko Andrey V. Logic Circuit Based Test Derivation for Microcontrollers //20th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2019: proceedings Erlagol, 29 june - 3 july 2019. Novosibirsk: NSTU publisher, 2019. P. 70-73, ISBN 978-1-7281-1753-9
15. Laputenko Andrey V., Timofey D. Petukhov, Nikolai A. Vasnev. Testing Microcontroller Based Physical Systems Using Finite Transition Models //19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2018: proceedings Erlagol, 29 june - 3 july 2018. Novosibirsk: NSTU publisher, 2018. P. 203-206, ISBN 978-1-5386-5020-2
16. Laputenko Andrey, Logic Circuit Based Test Derivation for Microcontrollers //20th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2019: proceedings Erlagol, 29 june - 3 july 2019. Novosibirsk: NSTU publisher, 2019. P. 70-73, ISBN 978-1-7281-1753-9
17. Laputenko Andrey, Timofey D. Petukhov, Nikolai A. Vasnev. Testing Microcontroller Based Physical Systems Using Finite Transition Models //19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2018: proceedings Erlagol, 29 june - 3 july 2018. Novosibirsk: NSTU publisher, 2018. P. 203-206, ISBN 978-1-5386-5020-2

18. Laputenko, A.V., López, J.E., Yevtushenko, N.V. Verifying Digital Components of Physical Systems: Experimental Evaluation of Test Quality // Russian Physics Journal. 2017. Vol. 60, № 11. P. 2012-2018. DOI: 0.1007/s11182-018-1317-6
19. López, J., Vinarsky E., Laputenko A. On the Fault Coverage of High-level Test Derivation Methods for Digital Circuits // EDM 2017: Proceeding of the 18th international conference on micro/nanotechnologies and electron devices, 2017. P. 184 – 189
20. López, J., and Maag, S. (2015). Towards a generic trust management framework using a machine-learning-based trust model. In 2015 IEEE Trustcom/BigDataSE/ISPA, volume 1, pp. 1343–1348.
21. López, J., Kushik, N., and Yevtushenko, N. (2017). Proactive trust assessment of systems as services. In Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering - Volume 1: ENASE, pp. 271–276.
22. López, J., Laputenko A., Kushik N., Yevtushenko N. and Torgaev S. (2018). Scalable Supervised Machine Learning Apparatus for Computationally Constrained Devices. In Proceedings of the 13th International Conference on Software Technologies - Volume 1:ICSOFT, ISBN 978-989-758-320-9, pages 518-528. DOI: 10.5220/0006908905520562.
23. McMillan K.L, “Interpolation and SAT-based model checking”. Proc. CAV ‘03, pp. 1-13, LNCS 2725, Springer, 2003.
24. Nica S. On the Use of Constraints in Program Mutations and its Applicability to Testing, PhD thesis, Graz Technical University. 2013.
25. Pomeranz I. and S. M. Reddy, "Pattern Sensitivity: A Property to Guide Test Generation for Combinational Circuits", in Proc. 8th Asian Test Symp., Nov. 1999, pp. 75-80.
26. Pomeranz I., S.M. Reddy. Test generation for multiple state-table faults in finite-state machines // IEEE Transactions on Computers, Vol. 46, No 7, pp. 782-794, 1997
27. Roth T. P. , W. G. Bouricius, P. R. Schneider, Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits. IEEE Transactions on Electronic Computers. EC-16, 567–580 (1967).
28. Smolov S.A., López J., Kushik N., Yevtushenko N., Chupilko M.M., Kamkin A.S. Testing logic circuits at different abstraction levels: An experimental evaluation // Proceedings of the IEEE East-West Design & Test Symposium, EWDTS, Yerevan, Armenia, 2016. pp. 189-192.
29. Test generation for Finite State Machine. URL: http://fsmtestonline.ru/
30. Villa, T., Yevtushenko, N., Brayton, R.K., Mishchenko, A., Petrenko, A., Sangiovanni-Vincentelli, A. The Unknown Component Problem. Theory and Applications. Springer, 2012, 312 p.
31. Vinarskii Evgenii, Andrey Laputenko, Jorge López, Natalia Kushik. Testing Digital Circuits: Studying the Increment of the Number of States and Estimating the Fault Coverage //19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices EDM 2018: proceedings Erlagol, 29 june - 3 july 2018. Novosibirsk: NSTU publisher, 2018. P. 220-224, ISBN 978-1-5386-5020-2
32. А.С. Твардовский, А.В. Лапутенко О возможностях автоматного описания параллельной композиции временных автоматов. Труды Института системного программирования РАН, том 30, вып. 1, 2018, стр. 25-40. DOI: 10.15514/ISPRAS-2018-30(1)-2.
33. Агибалов Г. П. Лекции по теории конечных автоматов. / Г. П. Агибалов, А. М. Оранов. – Томск : Издательство ТГУ, 1984. – 185 с.
34. Громов М.Л., Евтушенко Н.В., Лапутенко А.В. Использование временных автоматов при тестировании киберфизических систем// Изв. вузов. Физика. – 2016. – Т. 59. – № 12. – С. 174-176

35. Грунский И.С., Петренко А.Ф. Построение проверяющих экспериментов с автоматами, описывающими протоколы // Автоматика и вычислительная техника-1988, № 4.- С. 7-14.
36. Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П. Проектирование систем на микросхемах программируемой логики. — СПб.: БХВ -Петербург, 2002. — 608 с.
37. Карибский В. В. , П. П. Пархоменко, Г. Е. Сейтова, “О построении проверяющих последовательностей для дискретных устройств с памятью”, Автомат. и телемех., 1979, № 5, 167–177; Autom. Remote Control, 40:5 (1979), 764–772
38. Кондратьева О.В. Параллельная композиция конечных автоматов с таймаутами / О. В. Кондратьева, Н. В. Евтушенко, А. Р. Кавалли // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2014. – № 2. – С. 73–81.
39. Лапутенко А. В., Винарский Е. М. Синтез тестов для цифровых систем на высоком и низком уровнях абстракции // Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика. 2019. № 47. С.110–117. DOI: 10.17223/19988605/47/13.
40. Лапутенко А. В., Торгаев С.Н. Сравнение производительности различных самообучающихся моделей на примере задачи классификации // Новые информационные технологии в исследовании сложных структур: материалы 12-й международной конференции, 4 – 8 июня 2018 г. Томск: Издательский Дом ТГУ, 2018. С. 71-72.
41. Матросова А. Ю., Алгоритмические методы синтеза тестов. – Томск: Изд-во Томского университета, 1990. – 207 с
42. Пархоменко П. П., Согомонян Е. С. Основы технической диагностики. М.: Энергоиздат, 1981. 319 с.
43. Твардовский А.С, Лапутенко А.В. О возможностях автоматного описания параллельной композиции временных автоматов. Труды Института системного программирования РАН, том 30, вып. 1, 2018, стр. 25-40. DOI: 10.15514/ISPRAS-2018-30(1)-2.
44. Убар Р.Р. Проектирование контролепригодных дискретных систем: Учебное пособие. – Таллин: Таллинский политехнический институт, 1988. – 68 с.
45. Чегис И. А., Яблонский С. В. Логические способы контроля работы электрических схем. Тр. МИАН СССР, 51 (1958), с. 270—360

Краткий отчет

ПОЛУЧИТЬ ПОЛНЫЙ ОТЧЕТ

Лапутенко Автореферат НД

ПРОВЕРЕНО: 13.06.2020 11:37:29

№	Доля в отчете	Доля в тексте	Источник		Актуальна на	Модуль поиска	Блоков в отчете	Блоков в тексте
[01]	0,04%	6,7%	http://www.ispras.ru/uplo...		15 Сен 2018	Модуль поиска Интернет	3	30
[02]	3,1%	6,32%	Скачать полный выпуск (...		31 Июл 2018	Модуль поиска Интернет	8	26
[03]	0,79%	6,2%	О возможностях автомат...		28 Дек 2018	Модуль поиска Интернет	2	25

ЗАИМСТВОВАНИЯ
19,87%

САМОЦИТИРОВАНИЯ
0%

ЦИТИРОВАНИЯ
0%

ОРИГИНАЛЬНОСТЬ
80,13%

ИСТОЧНИКОВ: 20

ЕЩЕ НАЙДЕНО ИСТОЧНИКОВ: 17

ЗАИМСТВОВАНИЯ: 15,96%