

Чуркин Ян Андреевич

**Метод и программные средства статического анализа
описаний цифровой аппаратуры для выявления дефектов
проектирования**

Специальность 2.3.5 —
«Математическое и программное обеспечение вычислительных систем,
комплексов и компьютерных сетей»

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Работа выполнена в Федеральном государственном бюджетном учреждении науки Институте системного программирования им. В. П. Иванникова Российской академии наук.

Научный руководитель: **Бучацкий Рубен Артурович**,
кандидат технических наук

Официальные оппоненты: **Кореньков Владимир Васильевич**, доктор технических наук, старший научный сотрудник, научный руководитель Лаборатории информационных технологий им. М. Г. Мещерякова Объединенного института ядерных исследований

Чибисов Пётр Александрович, кандидат технических наук, старший научный сотрудник Федерального государственного бюджетного учреждения «Федеральный научный центр Научно-исследовательский институт системных исследований Национального исследовательского центра «Курчатовский институт»»

Ведущая организация: АО «МЦСТ»

Защита состоится 17 декабря 2026 г. в 15-00 на заседании диссертационного совета 24.1.120.01 при Федеральном государственном бюджетном учреждении науки Институте системного программирования им. В. П. Иванникова Российской академии наук по адресу: 115035, г. Москва, ул. Садовническая, д. 41, ст. 2.

С диссертацией можно ознакомиться в библиотеке и на сайте Федерального государственного бюджетного учреждения науки Института системного программирования им. В. П. Иванникова Российской академии наук.

Автореферат разослан «___» _____ 20__ г.

Ученый секретарь
диссертационного совета
24.1.120.01,

кандидат физико-математических наук

Турдаков Д. Ю.

Общая характеристика работы

Актуальность темы. Современные цифровые интегральные схемы характеризуются высокой сложностью, при этом к ним предъявляются жесткие требования по корректности, надежности и безопасности функционирования. На ранних этапах проектирования одним из основных уровней описания цифровых устройств является уровень регистровых передач (Register Transfer Level, RTL), на котором формализуются структура и поведение аппаратуры с использованием языков описания аппаратуры (Hardware Description Language, HDL), таких как Verilog, SystemVerilog, VHDL и других. Создание RTL-описания является этапом логического проектирования цифровой схемы. Под дефектом проектирования в работе понимается свойство RTL-описания, которое приводит или может привести к некорректному функционированию схемы, к расхождению ее поведения при имитационном моделировании и логическом синтезе либо к нарушению свойств безопасности. Корректность RTL-описаний во многом определяет качество и надежность конечной аппаратной реализации. Дефекты, внесенные на этом этапе, нередко выявляются лишь на более поздних этапах проектирования: в процессе функциональной верификации, логического синтеза или даже после изготовления микросхемы. Исправление таких дефектов сопровождается значительными временными и финансовыми затратами. При этом широко применяемые методы верификации на основе имитационного моделирования (simulation-based verification) не обеспечивают покрытия всех возможных состояний системы, особенно в условиях роста объема RTL-описаний и усложнения взаимодействий между компонентами цифровых систем.

В этих условиях особую значимость приобретает статический анализ RTL-описаний, позволяющий выявлять дефекты проектирования без имитационного моделирования и логического синтеза. Статический анализ должен выявлять на ранних стадиях разработки структурные и семантические дефекты, а также нарушения свойств безопасности. Последнее особенно важно с учетом возрастающего внимания к безопасности цифровой аппаратуры, нарушение которой может приводить к некорректной работе вычислительных систем или к возникновению аппаратных уязвимостей.

Методы статического анализа программ (построение абстрактных синтаксических деревьев, анализ потоков данных, контроль типов) хорошо изучены и широко применяются в компиляторах и инструментах анализа программно-го обеспечения. Однако их перенос на языки описания аппаратуры сопряжен с трудностями: такие языки описывают параллельно функционирующие компоненты с событийно-временной семантикой и четырехзначной логикой сигналов, сочетая поведенческие и структурные описания в одной модели, что усложняет построение внутренних представлений и моделей анализа.

Особую роль в RTL-описаниях играет управляющая логика, реализуемая в виде конечных автоматов. Недостижимые состояния и мертвые переходы,

тупиковые состояния и взаимоблокировки не выявляются синтаксическими проверками и требуют анализа графа состояний и переходов, а в современных RTL-описаниях управляющая логика состоит из взаимодействующих автоматов. Автоматы записываются в различных стилях кодирования, и распознавание по синтаксическим шаблонам часть из них пропускает. Поэтому метод статического анализа должен обеспечивать извлечение конечных автоматов из RTL-описаний независимо от стиля кодирования, их проверку и поиск взаимодействующих автоматов.

RTL-описание не содержит части сведений, необходимых для анализа схемы: устройства библиотечных ячеек, распределения питания и задержек. Эти сведения задают сопутствующие проектные форматы Liberty, UPF и SDF, и без их привлечения не выявляются, в частности, ошибки подсистемы питания. Поэтому метод статического анализа должен использовать информацию сопутствующих проектных описаний.

Задачей статического анализа RTL-описаний занимаются как промышленные, так и академические коллективы. Коммерческие анализаторы компаний Synopsys (SpyGlass), Cadence (JasperGold), Siemens EDA (Questa One) и Aldec (ALINT-PRO) реализуют тысячи правил и применяются в промышленности, однако закрыты для расширения, неполно поддерживают стандарты SystemVerilog IEEE 1800-2017 и IEEE 1800-2023 и интегрируют сопутствующие форматы лишь частично. Открытые средства либо не поддерживают SystemVerilog в полном объеме, либо ограничиваются проверками по синтаксическому дереву и развернутой модели без анализа значений сигналов, и ни одно из них не сочетает анализ потоков данных с полной поддержкой SystemVerilog. К ним относятся Verilator (W. Snyder), Verible и Surelog (альянс CHIPS), svlint, компилятор slang (M. Popoloski), инфраструктура логического синтеза Yosys (C. Wolf), платформа Qihe (лаборатория PASCAL) и инфраструктура CIRCT (проект LLVM). Академические группы Корнеллского университета (G. E. Suh, A. C. Myers, язык SecVerilog), Калифорнийского университета в Санта-Барбаре (T. Sherwood, B. Hardekopf, языки Caisson и Sapper), Калифорнийского университета в Сан-Диего (R. Kastner, Clepsydra) и Университета Флориды (F. Farahmandi, M. Tehranipoor, инструменты FSMx) сосредоточены на свойствах безопасности. Подходы групп Корнеллского и Калифорнийских университетов требуют аннотирования описания или перехода на расширенный язык, группа Университета Флориды извлекает автоматы из списков соединений вентильного уровня и RTL-описаний для оценки уязвимостей. Дефекты проектирования в стандартном SystemVerilog эти работы не затрагивают. В отечественных работах (С. А. Смоллов, А. С. Камкин, ИСП РАН) статический анализ RTL-описаний применялся для построения расширенных конечных автоматов в целях функциональной верификации. Таким образом, ни одно из существующих решений не сочетает полную поддержку SystemVerilog, расширяемый набор проверок, анализ конечных автоматов и интеграцию сопутствующих форматов.

В связи с этим актуальной является задача разработки и исследования метода статического анализа описаний цифровой аппаратуры, который опирается на набор правил, задающих проверяемые свойства описания, учитывает специфику языков описания аппаратуры при построении внутреннего представления, охватывает управляющую логику на уровне конечных автоматов и использует информацию сопутствующих проектных описаний. Метод должен быть реализован в виде расширяемой системы статического анализа, поддерживающей языки Verilog и SystemVerilog.

Целью диссертационной работы является разработка метода и программных средств статического анализа описаний цифровой аппаратуры на основе внутреннего представления проекта и набора правил, задающих проверяемые свойства описания, включая извлечение и анализ конечных автоматов и интеграцию сопутствующих проектных форматов, для выявления дефектов проектирования на ранних стадиях разработки цифровых интегральных схем.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Проанализировать существующие методы и программные средства статического анализа описаний цифровой аппаратуры и определить требования к методу анализа.
2. Разработать набор правил статического анализа, направленных на выявление структурных и семантических дефектов и нарушений свойств безопасности в описаниях цифровой аппаратуры, определив для каждого правила проверяемое свойство описания и сопоставив правилам техники анализа, достаточные для их проверки.
3. Разработать метод статического анализа описаний цифровой аппаратуры, основанный на внутреннем представлении проекта, обеспечивающий проверку разработанных правил без выполнения логического синтеза и имитационного моделирования.
4. Разработать метод извлечения и статического анализа конечных автоматов из RTL-описаний на основе сети логических элементов и графового представления для выявления независимого от стиля кодирования недостижимых состояний и мертвых переходов, тупиковых состояний и взаимоблокировок, а также взаимодействующих конечных автоматов с сопоставлением результатов анализа с исходным кодом.
5. Разработать метод интеграции сопутствующих проектных форматов (Liberty, UPF, SDF) в статический анализ на основе построения единой информационной модели.
6. Реализовать программные средства статического анализа в виде расширяемой системы в соответствии с разработанными методами и провести их экспериментальную оценку на открытых проектах цифровых СБИС,

включая проверку корректности реализованных детекторов, анализ выявляемых дефектов, а также сравнение полученных результатов с существующими открытыми и коммерческими инструментами статического анализа цифровой аппаратуры.

Научная новизна:

1. Разработаны метод статического анализа описаний цифровой аппаратуры на основе внутреннего представления проекта, сочетающий обход абстрактного синтаксического дерева (в том числе с языком сопоставления узлов), классификацию типов логики и анализ путей в процедурных блоках с интервальными ограничениями, анализ потоков данных с отслеживанием диапазонов значений, учет установленных разрядов и источников сигналов, анализ логической сети и распознавание семантических шаблонов, и набор правил для выявления структурных и семантических дефектов и нарушений свойств безопасности, в котором правилам сопоставлены техники анализа, достаточные для их проверки. Метод обеспечивает расширяемость набора проверок и раннее выявление дефектов без логического синтеза и имитационного моделирования.
2. Разработан метод извлечения и статического анализа конечных автоматов из RTL-описаний на основе сети логических элементов и графового представления, обеспечивающий независимо от стиля кодирования обнаружение недостижимых состояний и мертвых переходов, тупиковых состояний и взаимоблокировок, а также выявление взаимодействующих конечных автоматов с сопоставлением результатов анализа с исходным кодом.
3. Разработан метод интеграции сопутствующих проектных форматов (Liberty, UPF, SDF) в статический анализ описаний цифровой аппаратуры, основанный на построении единой информационной модели, связывающей развернутую модель описания с объектами сопутствующих описаний, включающий трансляцию библиотечных ячеек в поведенческое Verilog-представление и обеспечивающий статический анализ модели подсистемы питания (проверку необходимости, наличия и корректности стратегий изоляции и преобразования уровня напряжения), а также разбор и статическую проверку согласованности SDF-описания с развернутой моделью описания аппаратуры.

Теоретическая и практическая значимость. Теоретическая значимость работы заключается в разработке метода статического анализа описаний цифровой аппаратуры на основе внутреннего представления проекта, метода извлечения и статического анализа конечных автоматов из RTL-описаний на основе сети логических элементов и графового представления, а также метода интеграции сопутствующих проектных форматов в статический анализ. В работе определены понятия правила, нарушения и детектора, что позволяет сопоставить каждому правилу технику анализа, достаточную для его проверки.

Тупиковые состояния и взаимоблокировки конечного автомата определены через терминальные сильно связанные компоненты его графа. Сформулирован критерий необходимости стратегий изоляции и преобразования уровня напряжения по состояниям наборов питания. Метод на основе внутреннего представления может быть распространен на другие языки описания аппаратуры.

Практическая значимость работы заключается в реализации расширяемой системы статического анализа описаний цифровой аппаратуры на языках Verilog и SystemVerilog с поддержкой сопутствующих проектных форматов (Liberty, UPF, SDF), включающей набор детекторов правил, средства анализа конечных автоматов и анализатор модели подсистемы питания. Система позволяет выявлять дефекты проектирования в описаниях на языках Verilog и SystemVerilog на ранних стадиях разработки цифровых интегральных схем. Экспериментальная оценка системы проведена на открытых проектах цифровых СБИС в сравнении с коммерческим и открытыми инструментами, результаты подтверждают практическую применимость разработанных методов и их значимость для развития программного обеспечения САПР цифровых интегральных схем. Реализованная система статического анализа внедрена в научно-исследовательские и учебные проекты ИСП РАН, а также в опытно-конструкторские проекты АО «МНТЦ МИЭТ» для верификации описаний цифровой аппаратуры на ранних этапах проектирования. Переносимость архитектуры системы на другие языки описания аппаратуры подтверждена реализацией поддержки подмножества языка VHDL и построением единого абстрактного синтаксического дерева для описаний, сочетающих модули на языках Verilog и VHDL.

Методология и методы исследования. Результаты работы получены с использованием методов и моделей, применяемых при статическом анализе и верификации программ и аппаратных описаний. Математическую основу работы составляют теория графов, теория автоматов, теория формальных языков и теория множеств.

Соответствие паспорту научной специальности. Описания цифровой аппаратуры на языках Verilog, SystemVerilog и VHDL представляют собой программы на формальных языках со стандартизированной семантикой (IEEE 1364, IEEE 1800, IEEE 1076), поэтому разработанные методы являются методами статического анализа программ, учитывающими особенности этих языков. Область исследования соответствует паспорту научной специальности 2.3.5 «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» по пункту 1 «Модели, методы и алгоритмы проектирования, анализа, трансформации, верификации и тестирования программ и программных систем» (методы статического анализа описаний, извлечения и анализа конечных автоматов и интеграции сопутствующих проектных форматов, алгоритмы детекторов правил и проверки стратегий подсистемы питания) и по пункту 2 «Языки программирования и системы программирования, семантика программ» (анализ на основе семантики языков Verilog и SystemVerilog,

доработки компилятора SystemVerilog, язык сопоставления узлов абстрактного синтаксического дерева, интерпретатор языка UPF и фронтенд языка VHDL).

Основные положения, выносимые на защиту:

1. Метод статического анализа описаний цифровой аппаратуры на основе внутреннего представления проекта и набора из 118 правил для раннего выявления структурных и семантических дефектов и нарушений свойств безопасности без логического синтеза и имитационного моделирования.
2. Метод извлечения и статического анализа конечных автоматов из RTL-описаний на основе сети логических элементов и графового представления, выявляющий независимо от стиля кодирования недостижимые и тупиковые состояния, мертвые переходы, взаимоблокировки и взаимодействующие автоматы.
3. Метод интеграции сопутствующих проектных форматов в статический анализ на основе единой информационной модели, связывающей описание аппаратуры с объектами Liberty, UPF и SDF, для анализа схем с библиотечными ячейками, модели подсистемы питания и SDF-описаний.
4. Программные средства статического анализа описаний цифровой аппаратуры, реализующие предложенные методы, и результаты их экспериментальной оценки на открытых проектах цифровых СБИС, включая сравнение с открытыми и коммерческими инструментами.

Апробация работы. Основные результаты работы докладывались и обсуждались на научных семинарах Института системного программирования им. В. П. Иванникова РАН, а также на следующих научных конференциях и семинарах:

- Открытая конференция ИСП РАН, Москва, 11–12 декабря 2024 г.
- Конференция EDA Connect 2025: Bridging Industry and Academia, Ереван, Армения, 13–14 марта 2025 г.
- 67-я Всероссийская научная конференция МФТИ, Долгопрудный, 31 марта – 5 апреля 2025 г.
- Programming Languages and Compiler Technologies Summit 2025, Сочи, 20–22 апреля 2025 г.
- Международная конференция «Иванниковские чтения», Иркутск, 26–27 июня 2025 г.
- Открытая конференция ИСП РАН, Москва, 9–10 декабря 2025 г.
- Заседание №3 экспертно-аналитической группы «Аппаратная безопасность», Москва, 16 декабря 2025 г.
- Международная конференция DVCon China 2026 (Design and Verification Conference and Exhibition), Шанхай, КНР, 12–13 мая 2026 г.
- Международная конференция USBEREIT 2026 (IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology), Зеленоград, 14–15 мая 2026 г.

- Коллоквиум молодых ученых SYRCoSE 2026 (Spring/Summer Young Researchers' Colloquium on Software Engineering), Этномир, 28–31 мая 2026 г.
- Международная конференция ITC-Asia 2026 (International Test Conference in Asia), Нинбо, КНР, 19–21 августа 2026 г.

Личный вклад. Все результаты работы получены лично автором.

Публикации. По теме диссертации опубликовано восемь научных работ. Работы [1–4] опубликованы в журналах, входящих в перечень рецензируемых научных изданий ВАК. Работы [5–8] опубликованы в сборниках трудов научных конференций, причем работы [5; 6] опубликованы в изданиях, индексируемых международной базой цитирования Scopus. Получены два свидетельства о государственной регистрации программ для ЭВМ [9; 10].

В работах [1; 5] автору принадлежат разработка набора правил, метода статического анализа, анализа потоков данных с отслеживанием диапазонов значений, метода анализа конечных автоматов и архитектуры разработанной системы статического анализа, а также экспериментальная оценка. В работе [4] автором выполнен обзор методов и средств анализа, в работе [6] автором разработан метод извлечения и статического анализа конечных автоматов. В работах [2; 7] автору принадлежат разработка единой информационной модели и анализа стратегий питания, в работе [3] автору принадлежат разработка правил проверки синтезируемости и метода проверки пользовательских примитивов, в работе [8] автору принадлежат постановка задачи и проектирование языка сопоставления узлов абстрактного синтаксического дерева.

Объем и структура работы. Диссертация состоит из введения, шести глав, заключения, списка сокращений и условных обозначений, словаря терминов, списка литературы и четырех приложений. Полный объем диссертации составляет 197 страниц, включая 10 рисунков и 29 таблиц. Список литературы содержит 87 наименований.

Содержание работы

Во **введении** обоснована актуальность темы исследования, приведены сведения о степени ее разработанности, сформулированы цель и задачи работы, изложены научная новизна, теоретическая и практическая значимость, методология и методы исследования, положения, выносимые на защиту, а также сведения об апробации результатов и публикациях.

В **первой главе** приводится обзор методов и средств статического анализа описаний цифровой аппаратуры.

В разделе 1.1 выделены четыре особенности RTL-описаний как объекта статического анализа, отличающие их от программного кода: параллелизм процедурных блоков, событийно-временная семантика, четырехзначная логика сигналов (0, 1, X, Z) и совмещение поведенческого и структурного описания в одной модели.

В разделе 1.2 выявляемые дефекты разделены на структурные, семантические и нарушения свойств безопасности. Структурные дефекты (несоответствие разрядности сигналов, неподключенные порты, множественное управление сигналом) выявляются на уровне абстрактного синтаксического дерева (АСД). Семантические дефекты (некорректные типы присваиваний, неполные списки чувствительности, ошибки пересечения доменов синхронизации, проблемы сброса) требуют анализа с учетом тактовых сигналов. К нарушениям свойств безопасности отнесены, в частности, деление на ноль, выход за границы диапазонов и переполнения. Каждой группе дефектов поставлены в соответствие категории правил и техники анализа второй главы.

В разделе 1.3 рассмотрены методы статического анализа: обход АСД, анализ потоков данных, абстрактная интерпретация и символическое выполнение, извлечение и анализ конечных автоматов. Сопоставлены представления описаний, над которыми работают существующие средства анализа: синтаксическое дерево, развернутая модель, сеть логических элементов (описание схемы в терминах логических элементов и соединений между ними, которое строится средствами логического синтеза) и список соединений вентильного уровня. Ни одно из этих представлений не интегрирует сведения сопутствующих проектных форматов.

В разделе 1.4 рассмотрены сопутствующие проектные форматы. Формат Liberty описывает функции, временные характеристики и атрибуты тактовых сигналов библиотечных ячеек. Стандарт UPF (IEEE 1801) определяет домены питания, условия их переключения и стратегии изоляции. UPF-описание связано с RTL-описанием только через иерархию объектов, поэтому ошибки подсистемы питания анализом одного RTL-описания не выявляются. Формат SDF передает рассчитанные задержки в средства имитационного моделирования. Записи SDF-описания адресуются экземплярам, путям задержек и временным проверкам, поэтому согласованность SDF-описания с описанием аппаратуры можно проверить статически до имитационного моделирования.

В разделе 1.5 рассмотрен анализ пересечений доменов синхронизации и доменов сброса, разработка соответствующих детекторов выходит за рамки защищаемых положений. В разделе 1.6 методы на основе машинного обучения и больших языковых моделей рассмотрены как дополнение к детерминированному статическому анализу, так как они не дают гарантий полноты и корректности результата.

В разделе 1.7 рассмотрены коммерческие инструменты SpyGlass, JasperGold, ALINT-PRO, Questa One и открытые средства анализа и обработки RTL-описаний. Открытые средства сопоставлены по единой системе критериев (таблица 1). Показано, что единственным открытым синтаксическим анализатором, проходящим тесты набора sv-tests и обеспечивающим почти полную обработку синтаксиса SystemVerilog 2017 и 2023, является компилятор slang. Ни одно открытое средство не сочетает анализ потоков данных с полной

поддержкой SystemVerilog и не реализует полноценного анализа конечных автоматов. Совокупная интеграция форматов Liberty, UPF и SDF для целей анализа реализована преимущественно в коммерческих инструментах.

Таблица 1 — Сравнение открытых инструментов анализа и обработки RTL-описаний с разработанной системой

Инструмент	Поддержка SV 2017, 2023	Анализ потоков данных	Анализ FSM	Интерфейс правил	Интеграция форматов
Verible	частичная	нет	нет	нет	нет
svlint	почти полная	нет	нет	да	нет
Verilator	частичная	частично	нет	нет	нет
Surelog	почти полная	нет	нет	нет	нет
CIRCT	частичная	да (SSA)	нет	да	нет
Qihе	нет (Verilog)	да	нет	да	нет
Pyverilog	нет (Verilog)	да (DDG)	частично	нет	нет
slang	почти полная	частично	нет	да	нет
Icarus Verilog	частичная	нет	нет	нет	частично
Yosys	частичная	да	частично	да	частично
tree-sitter (Verilog)	нет (Verilog)	нет	нет	нет	нет
tree-sitter (SV)	частичная	нет	нет	нет	нет
SVAN (настоящая работа)	полная	да	да	да	да*

* Для Liberty и UPF реализована полная интеграция, для SDF реализованы разбор и проверка согласованности с моделью. Значение «частично» в столбце анализа FSM означает извлечение автоматов без анализа дефектов.

В разделе 1.8 сформулированы требования к методу статического анализа: поддержка Verilog и SystemVerilog в полном объеме синтезируемого подмножества, выявление структурных и семантических дефектов без синтеза и имитационного моделирования, проверка свойств безопасности, извлечение и анализ конечных автоматов, интеграция форматов Liberty, UPF и SDF, добавление правил без модификации ядра системы и работоспособность на промышленных RTL-описаниях объемом до миллиона строк. Сформулированы также требования к внутреннему представлению. Ни одно из рассмотренных представлений им не удовлетворяет, поэтому внутреннее представление строится как совокупность связанных уровней, привязанных к узлам развернутой модели.

Вторая глава посвящена методу статического анализа описаний цифровой аппаратуры на основе внутреннего представления.

В разделе 2.1 определено внутреннее представление, над которым выполняется анализ, и введены понятия правила, нарушения и детектора. Представление включает уровни, которые строит компилятор SystemVerilog: дерево разбора, АСД, предварительную и развернутую модели. В развернутой модели, построение которой называется элаборацией (elaboration), вычислены значения параметров, раскрыты блоки генерации, соединены порты и разрешены иерархические имена, что необходимо для анализа свойств, затрагивающих несколько модулей. Сами по себе уровни компилятора не отличаются от представлений других средств. Разработанное в работе представление дополняет их *внутренними структурами*, то есть вспомогательными структурами данных, которые строятся над развернутой моделью по заявкам детекторов и сохраняют привязку к ее узлам.

Правило r задает проверяемое свойство описания. Нарушением правила называется конкретное несоответствие описания этому свойству. Нарушение может относиться к отдельной конструкции, к отсутствию ожидаемой конструкции или к отношению между несколькими конструкциями. С каждым нарушением v связана локализация $\text{loc}(v)$, то есть диапазон в исходном тексте (имя файла, строка и столбец начала и конца фрагмента). Обозначим через P анализируемое описание, через $T(P)$ совокупность построенных по нему внутренних представлений, через $V_r(P)$ множество всех нарушений правила r . Детектором правила r называется алгоритм d_r , который по $T(P)$ вычисляет множество $\hat{V}_r(P)$ обнаруживаемых нарушений и формирует для каждого из них предупреждение

$$d_r(T(P)) = \{ w(v) \mid v \in \hat{V}_r(P) \}, \quad w(v) = (\text{loc}(v), r, m_r(v)),$$

где $m_r(v)$ обозначает диагностическое сообщение. В общем случае $\hat{V}_r(P)$ лишь аппроксимирует $V_r(P)$: элементы $\hat{V}_r(P) \setminus V_r(P)$ являются ложными срабатываниями, а элементы $V_r(P) \setminus \hat{V}_r(P)$ являются пропусками. Поэтому точность детекторов определяется экспериментально. Добавление правила сводится к реализации его детектора без модификации ядра анализа, что и обеспечивает расширяемость метода.

В разделе 2.2 сформулирован набор из 118 правил. Набор получен обобщением типовых проблем, собранных у инженеров дизайн-центров (АО «НТЦ «Атлас», АО «МНТЦ МИЭТ») и из опыта разработчиков. Каждая проблема уточнена до проверяемого свойства описания, при необходимости с явными исключениями, и сверена с каталогом STARC и наборами проверок коммерческих анализаторов. Набор классифицирован по 19 категориям и включает 114 правил для языков Verilog и SystemVerilog, 2 правила корректности модели подсистемы питания и 2 правила корректности пересечения доменов синхронизации и сброса. В работе реализована проверка 116 из 118 правил: 89 правил проверяются детекторами модуля анализа, 22 правила проверяются диагностиками компилятора, 3 правила о конечных автоматах проверяются подсистемой анализа автоматов, 2 правила модели питания проверяются анализатором модели подсистемы питания. Правила пересечения доменов синхронизации проверяются отдельным анализатором, разработка которого в работу не входит.

В разделах 2.3–2.7 описаны техники анализа и группы правил, к которым они применяются. Каждому правилу сопоставлена техника анализа, достаточная для его проверки. Техники различаются вычислительной стоимостью и множеством выявляемых свойств. Часть правил проверяется несколькими техниками совместно, поэтому сумма приведенных ниже чисел правил превышает число проверяемых правил.

- Обхода АСД достаточно для структурных правил, нарушение которых определяется по структуре описания без вычисления значений сигналов (65 правил, из них 22 проверяются диагностиками компилятора).
- Классификация типов логики процедурных блоков и анализ путей выполнения с интервальными ограничениями применяются к правилам

о типе логики и порядке операторов (22 и 5 правил). Пути не перечисляются: каждая ветвь обходится один раз, сведения о видах и порядке пройденных операторов объединяются в точках слияния ветвей, а интервальные ограничения отбрасывают несовместимые сочетания ветвей. Так, для правила, запрещающего в комбинационном блоке блокирующее присваивание после неблокирующего, состояние анализа хранит неблокирующее присваивание, которое могло встретиться до текущей точки, и ограничения ветвей, на которых оно встречено. Время обхода составляет $O(n \cdot q)$, где n обозначает число операторов блока, а q обозначает число выражений в условиях.

- *Распознавание семантического шаблона* (2 правила) рассмотрено на примере проверки синтезируемости пользовательских примитивов по их таблицам состояний.
- *Анализ на основе логической сети* (2 правила) применяется к правилам о замкнутых путях передачи значений. Сеть строится без синтеза на уровне объявлений и вхождений сигналов. Комбинационные петли выявляются выделением сильно связанных компонент алгоритмом Тарьяна и перечислением элементарных циклов только в отобранных компонентах.
- *Анализ потоков данных*, расширенный отслеживанием диапазонов значений сигналов, применяется к правилам о значениях сигналов и достижимости операторов, в том числе о делении на ноль и выходе за границы массива (7 правил).
- *Учет установленных разрядов и источников сигналов* в экземпляре и по иерархии применяется к правилам об инициализации, источниках и третьем состоянии сигналов (21 правило).

В разделе 2.8 для сокращения и унификации кода детекторов предложен предметно-ориентированный язык сопоставления узлов АСД SystemVerilog по образцу сопоставителей инструмента clang-tidy. Сопоставители записываются вложенными выражениями с агрегирующими конструкциями `allof`, `anyof`, `unless` и механизмом захвата узлов. Сопоставление выполняется в два этапа. Единственный полный обход дерева заполняет кэш, в котором каждому типу узла сопоставлены все узлы этого типа. Затем каждое правило просматривает только узлы своего верхнеуровневого типа, а тела модулей с несколькими экземплярами обходятся однократно. Детектор обратного порядка разрядов при подключении шин занимает на сопоставителях 16 строк против 33 строк на обходчиках, не считая общей функции сравнения диапазонов разрядов. В разделе 2.9 показано, что каждое правило настраивается независимо: его можно включить или отключить, задать уровень значимости, параметры и исключения, а также подавить диагностику локально в исходном тексте.

В третьей главе представлен метод извлечения и статического анализа конечных автоматов из RTL-описаний.

В разделе 3.1 на графе автомата определены дефекты (рисунок 1): мертвый переход, недостижимое состояние, тупиковое состояние (терминальная сильно связанная компонента из одного состояния) и взаимоблокировка (терминальная сильно связанная компонента из нескольких состояний). Терминальной называется сильно связанная компонента без исходящих переходов, не содержащая начального состояния автомата. В разделе 3.2 сравнение подходов к извлечению обосновало выбор сети логических элементов. Такая сеть обеспечивает независимость от стиля кодирования и естественную обработку распределенной и параметризованной логики, а потеря соответствия исходному коду устраняется разработанным механизмом сопоставления.

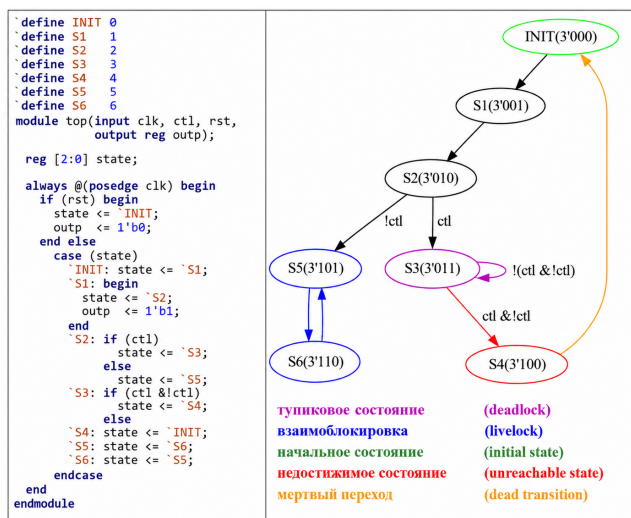


Рисунок 1 — Граф конечного автомата с отмеченными дефектами и соответствующее описание на языке SystemVerilog

В разделе 3.3 предложен метод извлечения автоматов из сети логических элементов. RTL-описание транслируется в представление RTLIL инфраструктуры логического синтеза Yosys, где автомат распознается по структуре схемы. Кандидатом на автомат считается триггер, вход данных которого формируется деревом мультиплексоров, на входах которых присутствуют только константы и выход того же триггера. В работе штатная схема извлечения инфраструктуры Yosys доработана. Частичное разделение проводов восстанавливает соответствие триггеров и проводов для массивов автоматов. Отбрасывание выходов, зависящих от сигналов помимо регистра состояния, позволяет единообразно обрабатывать автоматы Мили и Мура. Начальное состояние при синхронном сбросе определяется по числу достижимых состояний в поддеревьях корневого мультиплексора. Сложность извлечения в худшем случае составляет $O(|S| \cdot 2^{|I|} \cdot |M|)$, где $|S|$, $|I|$ и $|M|$ обозначают число состояний, число входов автомата и размер дерева мультиплексоров соответственно.

В разделе 3.4 дефекты выявляются анализом графа автомата (рисунок 1) в три этапа, каждый из которых уточняет граф для последующего. Мертвые переходы выявляются по противоречию условия перехода постоянным значениям входов автомата. Недостижимые состояния выявляются обходом в ширину от начального состояния без учета мертвых переходов. Тупиковые состояния и взаимоблокировки выделяются как терминальные среди сильно связанных компонент, вычисляемых алгоритмом Тарьяна за время $O(|V| + |E|)$. Анализ структурный: из графа удаляются только заведомо мертвые переходы, поэтому ни одно фактически достижимое состояние не объявляется недостижимым. Недостижимость, следующая лишь из соотношений между входами, не являющимися константами, может остаться невыявленной.

В разделе 3.5 результаты анализа сопоставляются с исходным кодом. Для этого в представлении RTLIL доработан формат атрибута исходного расположения (идентификатор источника и смещение в байтах), а имена состояний восстанавливаются распространением именованных констант по связанным присваиваниям символом до неподвижной точки. В разделе 3.6 два автомата считаются взаимодействующими, если сигнал, формируемый одним из них, достигает логики переходов другого по пути, содержащему произвольную комбинационную логику и не более одного последовательностного элемента. Взаимодействия выявляются прямым обходом сети логических элементов от каждого выхода автомата. В разделе 3.7 показано, что каждый экземпляр параметризованного автомата анализируется со своими значениями параметров, так как сеть логических элементов строится по развернутой модели. Так, в проекте BlackParrot состояния обратной записи контроллера кэша `br_use` достижимы в экземпляре для кэша данных и недостижимы в экземпляре для кэша команд, что соответствует архитектуре проекта.

Четвертая глава посвящена методу интеграции сопутствующих проектных форматов Liberty, UPF и SDF в статический анализ.

В разделе 4.1 рассмотрены компоненты подсистемы питания, описываемые в формате UPF (IEEE 1801): домены питания, сети питания и наборы сетей питания (далее наборы питания), состояния питания, ячейки изоляции, преобразователи уровня напряжения и регистры сохранения состояния. Выделены пять типов ошибок управления питанием: отсутствие или некорректная реализация преобразования уровня напряжения, изоляции домена и сохранения состояния, переход в недопустимое состояние питания и некорректная инициализация значений при включении питания. Ошибки первых двух типов выражаются свойствами модели подсистемы питания, поэтому их можно выявить статически. Ошибки остальных трех типов связаны с поведением схемы при переключении состояний питания и выявляются главным образом имитационным моделированием с учетом питания и формальной верификацией.

В разделе 4.2 описана единая информационная модель, на которой основан метод интеграции сопутствующих проектных форматов (рисунок 2). Единая информационная модель является частью внутреннего представления. Эта модель

связывает модули, экземпляры и порты описания аппаратуры с библиотечными ячейками Liberty, доменами питания и стратегиями UPF-описания и записями SDF-описания. Для всех трех форматов модель строится в три шага. Сначала по описанию строится объектная модель формата, затем ее объекты привязываются по именам к узлам развернутой модели, после чего над связанной моделью выполняются проверки, для которых недостаточно одного описания аппаратуры.



Рисунок 2 — Построение единой информационной модели из описания аппаратуры, библиотек Liberty, UPF- и SDF-описаний

Для межмодульного анализа разработан транслятор ячеек Liberty в поведенческое Verilog-представление, которое подключается к компилятору наравне с пользовательским описанием. По описанию последовательностной ячейки транслятор восстанавливает эквивалентное поведение триггера или защелки с сигналами синхронизации, сброса и установки. Экземпляры транслированных ячеек несут признак библиотечной ячейки, по которому модель подсистемы питания и детекторы отличают стандартные ячейки от пользовательской логики. Поддержка формата SDF (IEEE 1497-2001) реализована как статическая проверка согласованности SDF-описания с развернутой моделью описания аппаратуры по правилам раздела 32 стандарта IEEE 1800. Иерархические имена экземпляров разрешаются с учетом массивов экземпляров и блоков генерации. Записи задержек путей сопоставляются с путями блока `specify` по портам, фронту и канонической форме условия, а записи временных проверок сопоставляются с системными временными проверками. Рассогласования, которые система имитационного моделирования сопровождает лишь предупреждением, выявляются до имитационного моделирования.

В разделе 4.3 модель подсистемы питания строится полной интерпретацией UPF-описания по стандарту IEEE 1801-2018. Каждой команде стандарта сопоставлен обработчик, который проверяет аргументы и создает или изменяет объекты модели либо устанавливает их связь с объектами RTL-описания.

В разделе 4.4 разработан статический анализ стратегий изоляции и преобразования уровня напряжения (алгоритм 1). Проверка выполняется для каждого порта на границе соседних доменов. Стратегия изоляции необходима, если у набора питания источника сигнала есть состояние «выключено», а наборы питания источника и приемника не эквивалентны, то есть их сети не соединены между собой и не объявлены эквивалентными в UPF-описании. Стратегия преобразования уровня напряжения необходима, если наборы питания не эквивалентны и работают при разных номинальных напряжениях. Порт сопоставляется с фильтрами стратегий, заданных для этих доменов. Если ни одна стратегия порту не подходит, диагностика выдается по наиболее значимому невыполненному фильтру с указанием порта и строки UPF-описания.

Алгоритм 1 — Проверка стратегий изоляции и преобразования уровня напряжения на границах доменов питания

Вход: модель подсистемы питания: домены, наборы питания с состояниями, стратегии с наборами фильтров, вид проверяемой стратегии k

Выход: диагностики о нарушениях стратегий вида k

```

1   $P \leftarrow$  пары доменов, между элементами которых есть граничные порты;
2  для каждого пары доменов  $(D_1, D_2) \in P$  :
3      для каждого граничного порта  $p$  с набором питания источника  $a$ 
        и приемника  $b$  :
4          если  $p$  двунаправленный или не ТребуетсяСтратегия( $k, a, b$ ) то
            перейти к следующей итерации;
5           $S \leftarrow$  стратегии вида  $k$ , заданные для доменов  $D_1$  и  $D_2$ ;
6          если  $S = \emptyset$  то
7              выдать диагностику об отсутствии стратегии для порта  $p$ ;
8              перейти к следующей итерации;
9          для каждого стратегии  $s \in S$  : отметить фильтры  $s$ , которым  $p$ 
            не соответствует;
10         если нет стратегии, всем фильтрам которой  $p$  соответствует то
            выдать диагностику по наиболее значимому невыполненному фильтру;
11 функция ТребуетсяСтратегия( $k, a, b$ ):
12     если  $a$  и  $b$  эквивалентны то вернуть ложь;
13     если  $k$  является изоляцией то вернуть у  $a$  есть состояние «выключено»;
14     если заданы допустимые сочетания состояний  $a$  и  $b$  то вернуть в одном
        из них номинальные напряжения  $a$  и  $b$  положительны и различны;
15     вернуть множества номинальных напряжений  $a$  и  $b$  непусты и различны
        или одно из них содержит несколько значений;
```

В разделе 4.5 сведения библиотек Liberty применены в двух детекторах: детекторе RS-защелок, собранных из встроенных примитивов или библиотечных ячеек, и детекторе буферизованных тактовых сигналов. Связи ячеек Liberty с командами UPF устанавливаются при построении модели.

В **пятой главе** описана реализация разработанных методов в разработанной системе статического анализа.

В **разделе 5.1** описана архитектура разработанной системы статического анализа (рисунок 3). Система состоит из трех частей. Фронтенд выполняет разбор входных описаний: синтаксические анализаторы Verilog, SystemVerilog и подмножества VHDL строят деревья разбора, а интерпретаторы SDC и UPF, транслятор Liberty и анализатор SDF обрабатывают сопутствующие описания. Компилятор SystemVerilog на основе открытой библиотеки slang строит по деревьям разбора АСД и развернутую модель. Анализаторы выполняют проверки: модуль анализа svan-tidy содержит детекторы правил, подсистема svan-fsm на основе открытой инфраструктуры логического синтеза Yosys выполняет анализ конечных автоматов, анализатор модели подсистемы питания проверяет стратегии, а анализатор пересечений доменов синхронизации и сброса в работе не рассматривается.

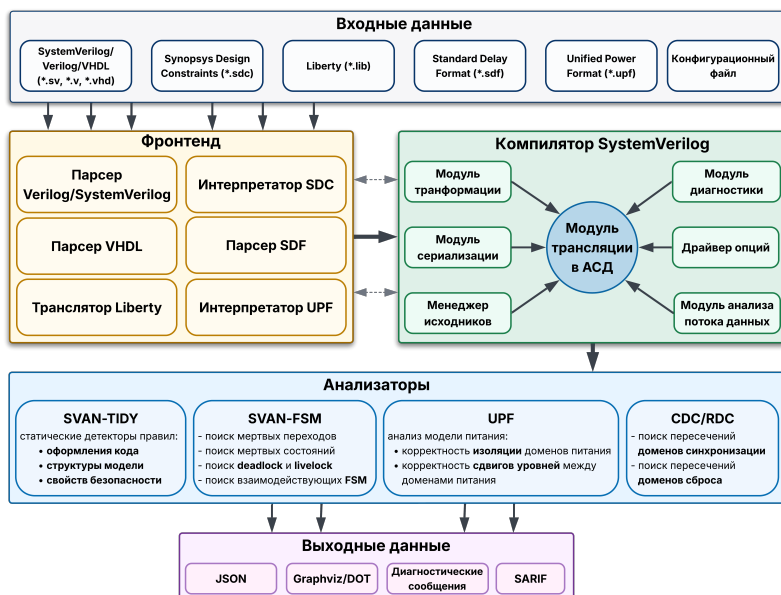


Рисунок 3 — Структурная схема системы статического анализа SVAN

Интерпретатор UPF построен на открытом интерпретаторе языка Tcl, в который добавлены команды стандарта IEEE 1801-2018. Для каждой из 63 команд разделов 6 и 7 стандарта реализован собственный обработчик. Кроме того, реализованы обработчики функций доступа к информационной модели UPF и команд запроса версии 2.0 стандарта, которые возвращают сведения о созданных объектах модели. Всего реализовано 120 обработчиков. Фронтенд VHDL транслирует конструкции языка в те же символы АСД, что и конструкции

SystemVerilog, поэтому построение развернутой модели, анализ потоков данных и детекторы применяются к ним без существенных изменений. Сущности и архитектуры VHDL отображаются в определения модулей, процессы отображаются в процедурные блоки со списками чувствительности, а обобщенные параметры отображаются в параметры модулей. В разделе 5.1.7 показано, что все детекторы модуля анализа реализуют общий интерфейс и объявляют, какие общие данные им нужны. Драйвер модуля объединяет заявки активированных детекторов и строит каждую структуру не более одного раза за запуск до начала работы детекторов.

В разделе 5.2 приведены выполненные в работе доработки открытых проектов, на которых построена система: компилятора slang, фронтенда yosys-slang, связывающего компилятор slang с инфраструктурой Yosys, и самой инфраструктуры Yosys. Существенная часть доработок принята в основные ветки этих проектов. В компиляторе slang реализованы проверка невырожденности последовательностей и разрешение тактовых событий утверждений с выявлением девяти классов ошибок, проверка покрытия таблиц состояний пользовательских примитивов, проверка дублирования разрядов в псевдонимах цепей и ряд исправлений синтаксического анализатора. Во фронтенде yosys-slang и инфраструктуре Yosys доработки обеспечили корректное построение сети логических элементов для извлечения конечных автоматов на открытых проектах. В разделе 5.3 приведены сведения о технических средствах. Разработанная система статического анализа реализована на языке C++ 20. Выходными данными системы являются диагностические предупреждения в текстовом формате или формате SARIF, АСД в формате JSON и графы конечных автоматов в формате DOT.

Шестая глава посвящена экспериментальной оценке разработанной системы статического анализа.

В разделе 6.1 изложена методика экспериментальной оценки. Тестовый набор составили десять проектов с открытым исходным кодом общим объемом свыше 1,8 млн строк (таблица 2): аппаратная реализация шифра AES, процессорные ядра RISC-V PicoRV32, SCR1, VeeR EH1, Ibex, VeeR EL2, BlackParrot, CVA6, OpenC910 и аппаратный корень доверия OpenTitan. Нагрузочный тест масштабируемости выполнен на проекте NVDLA объемом около 1,5 млн строк. Точность детекторов определялась на этом тестовом наборе. Срабатывания размечались вручную, точность вычислялась как доля истинных срабатываний $TP / (TP + FP)$. Если проверка дала на проекте не более 100 срабатываний, размечались все ее срабатывания на этом проекте, иначе размечались не менее 100 срабатываний, взятых по возможности из разных файлов. Точность разработанной системы на проекте вычислялась по разметке этого проекта. Полноту на открытых проектах измерить нельзя, так как число всех нарушений в них неизвестно. Поэтому полнота проверялась на синтетических тестах с заранее известными нарушениями. Для детекторов правил составлено 3134 синтетических и регрессионных теста (795 позитивных и 2339 негативных), для анализа

конечных автоматов составлено 49 тестов, для интерпретатора UPF составлено 830 тестов. Эксперименты выполнены на двух стендах. На сервере (восемь процессорных ядер 2,6 ГГц, 15 ГБ оперативной памяти, ОС CentOS 7) определялась точность детекторов правил и выполнялось сравнение со SpyGlass. На рабочей станции (процессор Intel Core i7-14700, 20 ядер, 31 ГБ оперативной памяти, ОС Ubuntu 24.04) выполнены сравнение с открытыми средствами, оценка механизма сопоставителей и измерение времени анализа конечных автоматов. Параллельное выполнение детекторов в экспериментах не использовалось: детекторы выполнялись последовательно в одном потоке.

В разделе 6.2 определена точность детекторов правил и диагностик компилятора разработанной системы статического анализа на тестовом наборе. При анализе всех исходных файлов десяти проектов, включая тестовое окружение, разработанная система выдала 53 486 предупреждений. Из них 40 322 относятся к шумному правилу PORT04 («порты должны быть регистровыми»), то есть к правилу с большим числом срабатываний, которое отражает практику проектирования, а не дефект. Все 100 размеченных срабатываний этого правила истинны, и при вычислении точности правило PORT04 не учитывается. Из остальных 13 164 предупреждений размечены 6846 (52 %), в том числе все предупреждения на шести проектах. Точность разработанной системы составила от **90,5** до **100 %** по проектам и **96,3 %** в целом (таблица 2).

Таблица 2 — Точность, время работы и пиковая память детекторов правил и диагностик компилятора разработанной системы статического анализа на проектах с открытым исходным кодом (без правила PORT04)

Проект	Строк, тыс.	Срабатываний	Размечено	FP	Точность	Время	Память
AES	4,6	4	4	0	100 %	0,6 с	35 МБ
PicoRV32	7,8	222	222	21	90,5 %	12 с	52 МБ
SCR1	16,9	377	377	9	97,6 %	17 с	4,7 ГБ
VeeR EH1	21,2	833	833	38	95,4 %	27 с	273 МБ
VeeR EL2	51,5	952	952	19	98,0 %	2,2 мин	1,7 ГБ
Ibex	85,0	234	234	1	99,6 %	22 с	438 МБ
BlackParrot	148,2	3980	977	13	99,7 %	25 с	782 МБ
CVA6	270,5	2543	1859	103	91,6 %	12,7 мин	7,7 ГБ
OpenC910	406,0	2132	726	2	99,9 %	8,3 мин	4,0 ГБ
OpenTitan	838,9	1887	662	99	91,1 %	24,2 мин	12,5 ГБ
Всего	1850,6	13 164	6846	305	96,3 %		

Точность по проекту получена переносом доли ложных срабатываний каждой проверки, определенной по размеченным срабатываниям этого проекта, на все срабатывания этой проверки в проекте.

Анализ крупнейших проектов занимает от 8 до 24 мин при пиковой памяти до 12,5 ГБ. Все 305 размеченных ложных срабатываний относятся к пяти проверкам. Детектор повторных присваиваний в одной области видимости (121 из 310 размеченных) сообщает о переопределении в условной ветви значения по умолчанию, о присваиваниях во взаимоисключающих ветвях

и о присваиваниях разным элементам массива на разных итерациях цикла, которые правило нарушением не считает. Диагностика компилятора о различной разрядности операндов (139 из 462) срабатывает на операндах одинаковой разрядности, которые различаются только знаковостью или перечислимым типом. Детектор несинтезируемых конструкций относит к ним синтезируемую системную функцию `$onehot0` (29 из 1127). Детектор неиспользуемых установленных битов (9 из 723) не считает чтением обращение к массиву по динамическому индексу, чтение выходного аргумента функции и чтение через модпорт интерфейса. Детектор усечения при присваивании (7 из 592) срабатывает на литерале `'bх` без размера, на значениях типа `chandle` и на результате унарного минуса той же разрядности. По результатам анализа разработчикам открытых проектов сообщено о 24 нарушениях. Разработчики подтвердили **13** нарушений, из них **10** исправлены, три ожидают исправления. К исправленным дефектам относятся выход за границы битового вектора в OpenTitan, выходы за границы массивов в CVA6 и недостижимый код в декодере команд ядра Ibex.

В разделе 6.3 подсистема анализа конечных автоматов разработанной системы статического анализа протестирована на 49 синтетических и регрессионных тестах и на десяти проектах тестового набора. В синтетических тестах подсистема нашла все внесенные дефекты четырех видов: мертвые переходы, недостижимые состояния, тупиковые состояния и взаимоблокировки. В десяти проектах подсистема извлекла 1934 экземпляра автоматов и выдала 1611 диагностик. Точность определена ручной разметкой всех извлеченных автоматов и диагностик, причем экземпляры одного автомата размечались один раз. Истинны 241 из 255 размеченных автоматов (**94,5 %**). Ложные извлечения относятся к регистрам, которые загружают значение по внешним условиям и удерживают его. Все размеченные диагностики указывают на действительные дефекты графа автомата. Точность анализа автоматов по всем диагностикам и извлеченным автоматам составила **95,5 %**. Большинство мертвых переходов и недостижимых состояний вызвано значениями параметров и константами конфигурации. Часть взаимоблокировок относится к многофазным автоматам, которые по замыслу разработчиков не возвращаются в начальное состояние. Меньшая часть диагностик указывает на потенциальные ошибки. Так, по мертвым переходам в проекте OpenC910 найден мертвый код, о котором сообщено разработчикам. Доработки извлечения, предложенные в разделе 3.3, более чем вдвое увеличивают число уникальных автоматов, извлекаемых из проекта OpenC910 (**88** против **34** при штатной схеме извлечения Yosys). Время анализа зависит от числа и сложности автоматов, а не от объема описания, что согласуется с оценкой сложности извлечения из раздела 3.3.

В разделе 6.4 интерпретатор UPF разработанной системы статического анализа проверен на 830 синтетических тестах языковых конструкций, а построение и анализ модели подсистемы питания проверены на 20 тестовых проектах, в том числе на контрольном проекте из приложения Е стандарта

IEEE 1801-2018. Открытых проектов с UPF-описаниями немного. Разработанная система проверена на UPF-описаниях проектов OpenC910, x-heep, 8b10b, Low-Power-Async-CDC-FIFO-IP и тестового экземпляра из проекта OpenROAD. В этих описаниях выявлены отклонения от стандарта, об отклонениях в четырех проектах сообщено разработчикам. Ошибка использования необъявленного объекта в экземпляре из OpenROAD подтверждена и исправлена разработчиками. Сравнение с инструментами VCS и SpyGlass показало, что оба инструмента поддерживают стандарт ограниченно, тогда как разработанная система поддерживает все команды IEEE 1801-2018. Анализатор SDF разработанной системы проверен на 94 модульных тестах, включающих все примеры раздела 32 стандарта IEEE 1800-2023, так как проекты тестового набора представляют собой RTL-описания и SDF-описаний не содержат. Детекторы на основе Liberty проверены на синтетических тестах. На проектах тестового набора срабатываний этих детекторов не зафиксировано. Эти проекты представляют собой RTL-описания до логического синтеза без экземпляров библиотечных ячеек, а RS-защелки из отдельных логических элементов и ручная буферизация тактовых сигналов характерны для описаний, адаптированных под конкретную технологическую библиотеку, и в тестовом наборе не встречаются.

В разделе 6.5 определено влияние механизма сопоставителей узлов АСД на время работы анализатора. Для этого три детектора разработанной системы статического анализа реализованы в двух вариантах: на сопоставителях и на обходчиках. Варианты на сопоставителях сократили полное время работы анализатора, включая разбор и построение развернутой модели, в **1,2–2,4** раза при запуске одного детектора и в **1,2–2,3** раза при совместном запуске трех детекторов. Всего в разработанной системе на сопоставителях реализовано 18 детекторов.

В разделе 6.6 разработанная система статического анализа сравнивается с коммерческим инструментом SpyGlass (версия V-2023.12) на десяти проектах тестового набора по 106 правилам набора, которым соответствуют 124 правила SpyGlass. Разработанная система завершила анализ всех десяти проектов, SpyGlass завершил анализ шести. Без шумных правил PORT04 и W287b точность разработанной системы по срабатываниям составила **95,7 %** против **79,8 %** у SpyGlass, с учетом этих правил точность составила **99,4** и **93,9 %** соответственно. Точность обоих инструментов вычислена по сопоставленным правилам набора. Из 124 сопоставленных правил SpyGlass сработали 23. Сообщения одного правила об одной позиции исходного текста (файл и строка) считаются одной находкой. На шести проектах, анализ которых SpyGlass завершил, разработанная система находит **49 %** истинных находок SpyGlass без учета шумного правила W287b (8 % с его учетом), а SpyGlass находит **29 %** находок разработанной системы без учета правила PORT04. Точность и взаимная полнота по проектам приведены в таблице 3. Все автоматы SpyGlass более чем с двумя состояниями найдены разработанной системой, тогда как SpyGlass обнаружил **7 %** автоматов, извлеченных разработанной системой. Детекторы правил разработанной

системы работают быстрее проверки сопоставленных правил в SpyGlass на всех шести проектах (24,2 против 111 мин на OpenTitan), хотя структурная проверка SpyGlass, к которой относятся 105 из 124 сопоставленных правил, занимает на OpenTitan 4,4 мин. Анализ автоматов разработанной системы на OpenTitan и OpenC910 выполняется медленнее, чем обнаружение автоматов в SpyGlass (83 и 177 мин против 4,2 и 4,0 мин).

Таблица 3 — Точность и взаимная полнота SVAN и SpyGlass по проектам

Проект	Срабатываний		Точность, %		Взаимная полнота, %	
	SVAN	SpyGlass	SVAN	SpyGlass	найдено SVAN	найдено SpyGlass
AES	4	4	100	50,0	0	0
PicoRV32	222	511	90,5	54,0	11	33
VeeR EH1	833	732	95,4	80,3	53	26
Ibex	234	191	99,6	75,9	54	36
OpenC910	2129	522	99,9	99,3	65	16
OpenTitan	1875	3860	91,1	80,7	49	45
Bcero	5297	5820	95,7	79,8	49	29

Без шумных правил и правил об автоматах. Столбец «найдено SVAN» содержит долю истинных находок SpyGlass, найденных разработанной системой, столбец «найдено SpyGlass» содержит обратную долю.

В разделе 6.7 разработанная система статического анализа сравнивается с открытыми средствами Verilator, svlint, Verible, Surelog и Yosys по 42 правилам, которые проверяются хотя бы одним открытым средством. Точность разработанной системы по этим правилам составила **94,5 %**. В сравнении с Surelog, который проверяет три правила набора, точность обоих средств близка (**97,2** и **99,5 %**). Точность Verilator, svlint и Verible, вычисленная по сопоставленным правилам набора, ниже (**38,8**, **45,9** и **34,2 %**). Точность извлечения автоматов у разработанной системы и Yosys близка (**91,7** и **89,1 %**). Разработанная система находит от **62** до **100 %** истинных находок каждого открытого средства, а открытые средства находят от **14** до **50 %** находок разработанной системы. Кроме того, разработанная система охватывает области, которые не покрывает ни одно из рассмотренных открытых средств: модель подсистемы питания, правила на основе библиотек Liberty, поиск тупиковых состояний, недостижимых состояний и взаимоблокировок автоматов. При этом детекторам правил разработанной системы на рабочей станции требуется до 12,7 ГБ памяти и до 9,1 мин.

В разделе 6.8 приведены сведения о внедрении. Разработка выполнена в рамках опытно-конструкторской работы «Разработка системы статического анализа для языка описания аппаратуры» (шифр «САПР-Анализ»). На основе результатов работы подготовлена дополнительная профессиональная программа повышения квалификации «Применение системы статического анализа SVAN для языка описания аппаратуры».

В **приложениях** приведены каталог правил с указанием техники проверки каждого правила, примеры выявленных дефектов и перечень обращений

к разработчикам, материалы сравнения с открытыми средствами и сведения о внедрении и регистрации результатов.

В **заключении** сформулированы основные результаты работы.

1. Разработан метод статического анализа описаний цифровой аппаратуры на основе внутреннего представления проекта и набора из 118 правил для раннего выявления структурных и семантических дефектов и нарушений свойств безопасности без логического синтеза и имитационного моделирования.
2. Разработан метод извлечения и статического анализа конечных автоматов из RTL-описаний на основе сети логических элементов и графового представления, выявляющий независимо от стиля кодирования недостижимые и тупиковые состояния, мертвые переходы, взаимоблокировки и взаимодействующие автоматы.
3. Разработан метод интеграции сопутствующих проектных форматов в статический анализ на основе единой информационной модели, связывающей описание аппаратуры с объектами Liberty, UPF и SDF, для анализа схем с библиотечными ячейками, модели подсистемы питания и SDF-описаний.
4. На основе предложенных методов реализованы программные средства статического анализа описаний цифровой аппаратуры, включающие детекторы правил разработанного набора, средства анализа конечных автоматов и анализатор модели подсистемы питания. Экспериментальная оценка на открытых проектах цифровых СБИС и сравнение с коммерческим и открытыми инструментами статического анализа показали эффективность разработанных методов.

К дальнейшим направлениям развития относятся расширение набора правил и поддержка стандартов кодирования (например, STARC), поддержка версий стандарта UPF, отличных от IEEE 1801-2018, завершение поддержки языка VHDL и исследование гибридных схем, сочетающих детерминированный статический анализ с методами машинного обучения.

Публикации автора по теме диссертации

1. Система статического анализа для языка описания аппаратуры SystemVerilog [Текст] / Я. А. Чуркин, Р. А. Бучацкий [и др.] // Труды Института системного программирования РАН. — 2025. — Т. 37, № 1. — С. 7—40. — (БАК).
2. Статический анализ модели подсистемы питания цифровой аппаратуры [Текст] / А. В. Егиазарян, Я. А. Чуркин [и др.] // Труды Института системного программирования РАН. — 2026. — Т. 38, № 3. — С. 45—70. — (БАК).

3. Автоматическая проверка синтезируемости описаний аппаратуры [Текст] / А. А. Шевцов, Я. А. Чуркин [и др.] // Труды Института системного программирования РАН. — 2026. — Т. 38, № 4, ч. 2. — С. 45—66. — (БАК).
4. Чуркин, Я. А. Статический анализ описаний цифровой аппаратуры: методы, инструменты и перспективы [Текст] / Я. А. Чуркин, Р. А. Бучацкий // Программные продукты и системы. — 2026. — Т. 39, № 2. — С. 264—274. — (БАК).
5. SVAN: Static Analysis System for SystemVerilog Hardware Descriptions [Text] / Y. Churkin, R. Buchatskiy, [et al.] // 2026 IEEE Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (US-BEREIT). — Yekaterinburg, Russian Federation : IEEE, 2026. — P. 1—4. — (Scopus).
6. SVAN-FSM: Static Analysis of Finite State Machines in SystemVerilog [Text] / A. Viazovtsev, Y. Churkin, [et al.] // 2026 IEEE International Test Conference in Asia (ITC-Asia). — Ningbo, China : IEEE, 2026. — P. 1—6. — (Scopus).
7. Китаев, К. Н. Разработка средств спецификации и верификации подсистемы питания в статическом анализаторе для языка описания аппаратуры SystemVerilog [Текст] / К. Н. Китаев, Я. А. Чуркин // Труды 67-й Всероссийской научной конференции МФТИ. Радиотехника и компьютерные технологии. — М. : Физматкнига, 2025. — С. 31—32.
8. Трунов, В. В. Разработка предметно-ориентированного языка для сопоставления узлов абстрактного синтаксического дерева языка описания аппаратуры SystemVerilog [Текст] / В. В. Трунов, Я. А. Чуркин // Труды 67-й Всероссийской научной конференции МФТИ. Радиотехника и компьютерные технологии. — М. : Физматкнига, 2025. — С. 42—43.
9. Свидетельство о государственной регистрации программы для ЭВМ № 2026615835 Рос. Федерация. Система статического анализа для языка описания аппаратуры [Текст] / Я. А. Чуркин [и др.] ; Российская Федерация, от имени которой выступает Министерство промышленности и торговли Российской Федерации; АО «МНТЦ МИЭТ». — № 2025692899 ; заявл. 26.11.2025 ; опубл. 03.03.2026, Бюл. № 3 (Рос. Федерация).
10. Свидетельство о государственной регистрации программы для ЭВМ № 2026683339 Рос. Федерация. Система статического анализа описаний аппаратуры на языке VHDL [Текст] / Я. А. Чуркин [и др.] ; Федеральное государственное бюджетное учреждение науки Институт системного программирования им. В.П. Иванникова Российской академии наук. — № 2026681213 ; заявл. 09.07.2026 ; опубл. 30.07.2026 (Рос. Федерация).

Чуркин Ян Андреевич

Метод и программные средства статического анализа описаний цифровой аппаратуры
для выявления дефектов проектирования

Автореф. дис. на соискание ученой степени канд. техн. наук

Подписано в печать _____._____._____. Заказ № _____

Формат 60×90/16. Усл. печ. л. 1,63. Тираж 100 экз.

Типография _____