ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ, ԳԻՏՈՒԹՅԱՆ,
ՄՇԱԿՈՒՅԹԻ ԵՎ ՍՊՈՐՏԻ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Մանուկյան Արման Գևորգի

ԹՎԱՅԻՆ ԻՆՏԵԳՐԱԼ ՍԽԵՄԱՆԵՐԻ՝ ԲԱՐՁՐ ԱՐԱԳԱԳՈՐԾՈՒԹՅԱՄԲ ՍՏՈՒԳՄԱՆ ՄԻՋՈՑՆԵՐԻ ՄՇԱԿՈՒՄԸ

Ե.27.01 «Էլեկտրոնիկա, միկրո և նանոէլեկտրոնիկա» մասնագիտությամբ
տեխնիկական գիտությունների թեկնածուի գիտական աստիճանի
հայցման ատենախոսության

ՍԵՂՄԱԳԻՐ

Երևան 2025

МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ, КУЛЬТУРЫ И СПОРТА
РЕСПУБЛИКИ АРМЕНИЯ

НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ АРМЕНИИ

Манукян Арман Геворгович

РАЗРАБОТКА БЫСТРОДЕЙСТВУЮЩИХ СРЕДСТВ ТЕСТИРОВАНИЯ ЦИФРОВЫХ ИНТЕГРАЛЬНЫХ СХЕМ

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата
технических наук по специальности 05.27.01-
“Электроника, микро- и наноэлектроника”

Ереван 2025

Ատենախոսության թեման հաստատվել է Հայաստանի ազգային պոլիտեխնիկական համալսարանում (ՀԱՊՀ)։

Գիտական ղեկավար՝ տ.գ.դ. Օլեգ Հարությունի Պետրոսյան

Պաշտոնական ընդդիմախոսներ՝ տ.գ.դ. Ռուբեն Ռաֆայելի Վարդանյան
տ.գ.թ. Գոռ Արշավիրի Աբգարյան

Առաջատար կազմակերպություն՝ Հայ-ռուսական համալսարան

Ատենախոսության պաշտպանությունը կայանալու է 2025թ. հուլիսի 24-ին, ժամը 12^{00}-ին, ՀԱՊՀ-ում գործող «Ռադիոտեխնիկայի և էլեկտրոնիկայի» 046 մասնագիտական խորհրդի նիստում (հասցեն՝ 0009, Երևան, Տերյան փ., 105, 17-րդ մասնաշենքի)։

Ատենախոսությանը կարելի է ծանոթանալ ՀԱՊՀ–ի գրադարանում։

Սեղմագիրն առաքված է 2025թ. հունիսի 16-ին

046 Մասնագիտական խորհրդի գիտական քարտուղար, տ.գ.թ. Բենիամին Ֆելիքսի Բաղալյան

Тема диссертации утверждена в Национальном политехническом университете Армении (НПУА)

Научный руководитель: д.т.н. Олег Арутюнович Петросян

Официальные оппоненты: д.т.н. Рубен Рефаелович Варданян
к.т.н. Гор Аршавирович Абгарян

Ведущая организация: Росссийско-армянский университет

Защита диссертации состоится 24-го июля 2025 г. в 12^{00} ч. на заседании Специализированного совета 046 — "Радиотехники и электроники", действующего при Национальном политехническом университете Армении, по адресу: 0009, г. Ереван, ул. Теряна, 105, корпус 17.

С диссертацией можно ознакомиться в библиотеке НПУА.

Автореферат разослан 16-го июня 2025г.

Ученый секретарь Специализированного совета 046, к.т.н. Бениамин Феликсович Бадалян

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. Проверка цифровых схем играет ключевую роль во всём процессе проектирования, обеспечивая функциональную точность и надёжность. С развитием современных интегральных схем (ИС) увеличивается не только их функциональная сложность, но и количество электронных элементов в них, достигая нескольких миллиардов. Рост функциональной сложности ИС и числа логических компонентов в схемах, в свою очередь, приводит к необходимости проектирования тестовой среды с более высоким уровнем покрытия и увеличению времени, затрачиваемого на процесс верификации.

Для достижения максимального значения покрытия верификации необходимо обеспечить не только все возможные комбинации входных значений схемы, но и их возможные последовательности. Недостатки функционального покрытия могут привести к невыявленным дефектам ИС, что, в свою очередь, может вызвать сбои или отказы системы. По этой причине в современных процессах верификации широко применяются среды, управляемые покрытием. Верификация, управляемая покрытием, позволяет моделировать входные последовательности на основе уже покрытых входных значений. Проектирование и моделирование такой среды требуют значительных временных затрат. Именно поэтому в последние годы 70–80% всего процесса проектирования схем уходит на этап верификации. Следовательно, разработка методов быстродействующей верификации цифровых схем имеет большое значение.

Поскольку в современных вычислительных системах интеллектуальная собственность памяти (ИСП) составляет около 70% от общего объёма, корректная работа таких блоков крайне важна. В современных ИС блоки ИСП не являются изолированными единицами, а тесно интегрированы с другими функциональными блоками. Логические блоки, включая комбинаторные и последовательностные, часто используют память для временного хранения состояний конечных автоматов.

Кроме того, память играет важную роль во взаимодействии с внешними компонентами. Блоки ввода/вывода могут использовать память для буферизации данных перед их передачей другим системам, а периферийные блоки, такие как датчики или модули связи, применяют память для хранения данных с датчиков или управления буферами связи. Учитывая, что ошибки в блоках ИСП могут существенно повлиять на работу, функциональность и надёжность ИС, необходима тщательная верификация блоков ИСП.

С прогрессом новейших технологий роль управляющих блоков статических оперативных запоминающих устройств (СОЗУ) становится ещё более значимой в высокопроизводительных вычислительных системах. Управляющий блок СОЗУ выступает в качестве моста между центральным процессором и СОЗУ, обеспечивая эффективную передачу данных. В современных ИС, где используются многоядерные центральные процессоры, управляющие блоки СОЗУ должны обеспечивать одновременный доступ к различным модулям памяти по разным потокам, одновременно управляя конфликтами. Такие управляющие блоки имеют сложную архитектуру, и обеспечение их качества также требует проектирования сложной среды верификации, а также длительной и детальной проверки.

Объект исследования. Методы повышения скорости автоматизированного проектирования и верификации сред функциональной проверки цифровых ИС.

Цель работы. Разработка методов автоматизированного проектирования быстродействующих сред с высоким охватом проверки цифровых ИС.

Методы исследования. В ходе исследований использовались подходы к созданию и моделированию верификационных сред, современные методы оценки и улучшения их рабочих характеристик, а также теории эффективного применения процесса верификации и его компонентов.

Научная новизна:

- Предложен метод разработки быстродействующих средств верификации многопортовых СОЗУ на основе применения агентов УВМ и параллельного моделирования, что позволило сократить время моделирования на 20%, а объем используемой памяти - на 20...30%. Однако время, затраченное на проектирование, увеличилось на 50%.
- Разработан метод быстродействующей верификации блока управления СОЗУ, работающего по УВМ, который за счёт использования очередей обеспечивает высокий уровень покрытия при сокращении времени моделирования на 25% и уменьшении обьема используемой памяти на 20...35%. Однако проектирование занимает на 30...45% больше времени.
- Разработан метод автоматической генерации быстродействующих верификационных сред цифровых ИС на основе новых библиотек, созданных с использованием языка Python. Применение метода сокращает время проектирования на 40–50%.
- Создано программное средство (ПС) для автоматизированного проектирования цифровых ИС и их верификационных сред, которое позволяет сократить общее время верификации в 3...5 раз за счёт увеличения обьема используемой памяти на 15%.

Практическая ценность работы. Разработан ПС DGV для автоматического проектирования быстродействующих средств верификации цифровых ИС, который позволяет создавать функциональные верификационные среды на основе современных методик с внедрением предложенных методов. Использование ПС в проектировании и проверке СОЗУ позволяет сократить время проектирования в 5 раз, а для их управляющих блоков — примерно в 3 раза, обеспечивая при этом создание быстродействующих верификационных сред с различными конфигурациями интеллектуальной собственности при увеличении использования памяти на 15%.

На защиту выносятся:

- метод разработки быстродействующих средств верификации многопортовых СОЗУ на основе параллельного моделирования с применением дополнительных агентов.
- метод разработки верификационных средств для блока управления СОЗУ с высоким покрытием путём использования очередей.
- Метод автоматической генерации верификационных сред для СОЗУ и их управляющих блоков с использованием новых библиотек на языке программирования Python.

• ПС для проектирования цифровых ИС и их быстродействующих верификационных решений.

Достоверность научных положений Достоверность научных положений подтверждена экспериментальными результатами моделирования и математическими обоснованиями, представленными в диссертации.

Внедрение. ПС Design Generator and Validator (DGV) был внедрён в компании Xilinx Armenia. Он предназначен для автоматизированного проектирования цифровых схем и соответствующих верификационных сред. Применение разработанного инструмента DGV значительно упростило процесс проверки цифровых схем и сократило время, затрачиваемое на верификацию. Кроме того, программное средство было интегрировано в процесс проектирования FPGA в компании Xilinx Armenia, что позволило упростить весь проектный цикл.

Апробация работы. Основные научные и практические результаты диссертации докладывались на:

- 19-й Международной конференции "East-West Design & Test Symposium (EWDTS)" (Батуми, Грузия, 2023 г.);
- научных семинарах кафедры "Электроника, биомедицинские и измерительные системы" Национального политехнического университета Армении - НПУА (Ереван, Армения, 2022–2025 гг.);
- научных семинарах кафедры "Микроэлектронные схемы и системы" НПУА (Ереван, Армения, 2022–2025 гг.)
- научных семинарах компании "Xilinx Armenia" (Ереван, Армения, 2023–2025 гг.).

Публикации. Основные положения диссертации представлены в пяти научных работах, список которых приведен в конце автореферата.

Структура и объем диссертации. Работа состоит из введения, трех глав, основных выводов, списка литературы из 123 наименований и 5-и приложений. В первом приложении представлен акт внедрения диссертации, во втором - фрагмент верификационной среды для многопортовой СОЗУ, в третьем -описание разработанного ПС DGV, в четвертом - перечень использованных таблиц, рисунков и а в пятом - список используемых аббревиатур. Объем диссертации составляет 111 страниц, а вместе с приложениями – 168 страницы. Десетрация написана на армянском языке.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность темы диссертации, сформулированы цель и основные задачи исследования, представлены методы исследования, научная новизна, практическое значение и основные научные положения, выносимые на защиту.

В первой главе представлены реализованные методы тестирования цифровых ИС, СОЗУ и их блоков управления.

По мере усложнения функциональных возможностей современных ИС увеличивается и вероятность возникновения функциональных дефектов в ИС. По этой причине функциональное тестирование проводится после каждого этапа проектирования ИС, на которое приходится около 80% времени, затрачиваемого на весь проект (рис. 1). В результате разработка высокопроизводительных сред тестирования ИС стала одной из наиболее актуальных проблем.

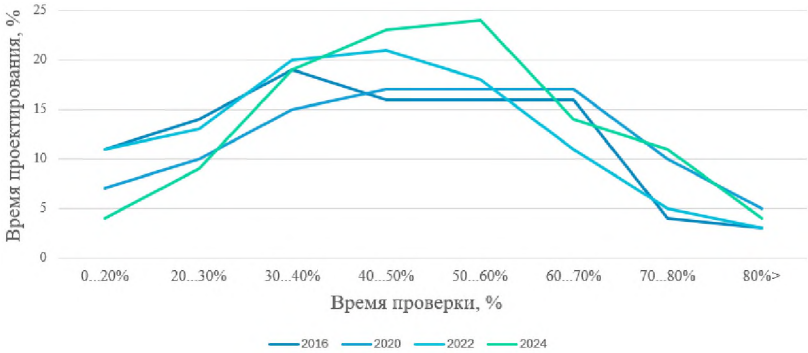

Рис. 1. Время, затрачиваемое на процесс проверки и проектирования

Многопортовые СОЗУ. Интеллектуальные свойства СОЗУ используются во всех компонентах ИС. Поэтому при верификации конструкций других компонентов необходимо моделировать интеллектуальные свойства взаимодействующих с ними ЗУ с использованием языков описания аппаратуры (HDL). Современные маршрутизаторы применяют многопортовые СОЗУ, которые, в отличие от однопортовых, позволяют осуществлять одновременный доступ к нескольким адресам (рис. 2).

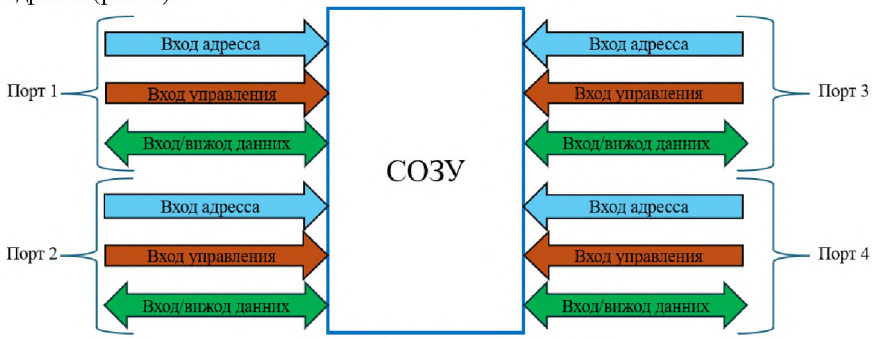

Рис. 2. Многопортовое СОЗУ

Из этого следует, что при верификации необходимо учитывать проверку операций записи и чтения для всех портов по всем адресам СОЗУ. Формула покрытия для многопортового ЗУ может быть представлена следующим образом:

$$C_{multiport}^{cross-check} = \frac{|A_{accessed} \cap O_{executed} \cap P_{used} \cap C_{tested}|}{|A_{total} \times O_{total} \times P_{total} \times C_{total}|} \times 100\% \quad (1)$$

Современные методы тестирования многопортовых СОЗУ используют методологию UVM, обеспечивающую повторное использование среды, генерацию случайных сценариев и эффективное выявление ошибок с помощью покрытия. (рис. 3).

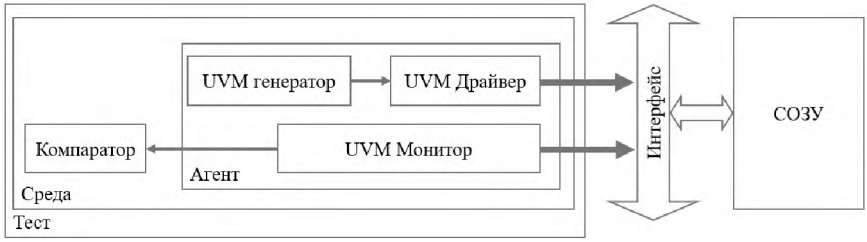

Рис. 3. Предлагаемая тестовая среда для многопортового СОЗУ

Несмотря на указанные преимущества, использование UVM сопряжено с рядом недостатков. Разработка среды на основе UVM требует глубоких знаний как в области цифровой схемотехники, так и в области объектно-ориентированного программирования. Кроме того, это увеличивает время моделирования и объём памяти, используемой в процессе тестирования. С использованием этого метода были проведены различные имитационные эксперименты для оценки времени, затрачиваемого на проверку, и используемой памяти (табл. 1).

Таблица 1

Время, затрачиваемое на тестирование и моделирование многопортового СОЗУ, и используемая память

Параметр	Количество портов для ОЗУ		
	2	3	4
При проектировании тестовой среды (*ч*)	4	5	6
Время моделирования (м)	32	40	49
Использованная память (Гб)	2	2,6	3

Контроллер памяти. С развитием современных технологий роль блоков управления СОЗУ становится всё более значимой в высокопроизводительных вычислительных системах. Эти блоки часто содержат встроенные помехоустойчивые модули кодирования, обеспечивающие надёжную запись и считывание данных, такие как модули контроля чётности и коррекции ошибок (рис. 4).

Рис. 4. Контроллер памяти с механизмом исправления ошибок кода

Для тестирования таких блоков управления необходимо разработать среду, которая позволяет принудительно генерировать ошибки в данных во время их записи (рис. 5).

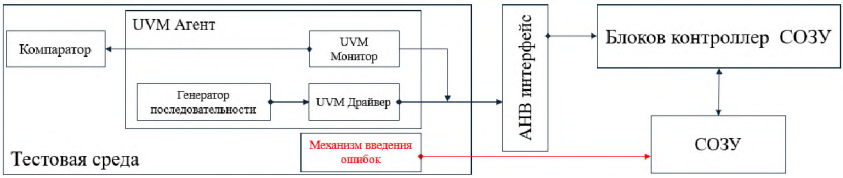

Рис. 5. Предлагаемая тестовая среда блока управления ОЗУ с помехозащищенными блоками кодирования

В зависимости от алгоритмов обнаружения и коррекции ошибок следует предусмотреть все возможные комбинации битовых искажений. Эти комбинации определяются по следующей формуле:

$$\binom{m}{k} = \frac{m!}{k!(m-k)!} \qquad (2)$$

Для оценки времени моделирования и объёма используемой памяти были проведены имитационные эксперименты на контрольных блоках SoC с различной разрядностью, используя существующий метод (табл. 2–4).

Таблица 2

Время и память, затрачиваемое на моделирование 8-битного контроллера СОЗУ

Количество искаженных битов	Время проектирования (*ч*)	Время симуляции (*ч*)	Используемая память (*Гб*)
1	7	2	3
2	10	4	5

Таблица 3

Время и память, затрачиваемое на моделирование 16-битного контроллера СОЗУ

Количество искаженных битов	Время проектирования (*ч*)	Время симуляции (*ч*)	Используемая память (*Гб*)
1	7	6	5
2	10	12	8

Таблица 4

Время и память, затрачиваемое на моделирование 32-битного контроллера СОЗУ

Количество искаженных битов	Время проектирования (*ч*)	Время симуляции (*ч*)	Используемая память (*Гб*)
1	7	8	7
2	10	16	12

Существующие методы создания высокопроизводительных тестовых сред для цифровых ИС. Существующие методы создания высокопроизводительных тестовых сред для цифровых ИС. Использование автоматизированных генераторов тестовых сред стало насущной необходимостью при проектировании и тестировании современных цифровых ИС. Поскольку значительная часть времени верификации

уходит на разработку тестовой среды, возникает потребность в автоматизации этого процесса.

В настоящее время разработаны различные методы, направленные на улучшение процедуры проверки цифровых схем (рис. 7). Несмотря на то, что эти методы в определённой степени повысили эффективность верификации, они всё ещё имеют недостатки с точки зрения удобства использования и масштабируемости..

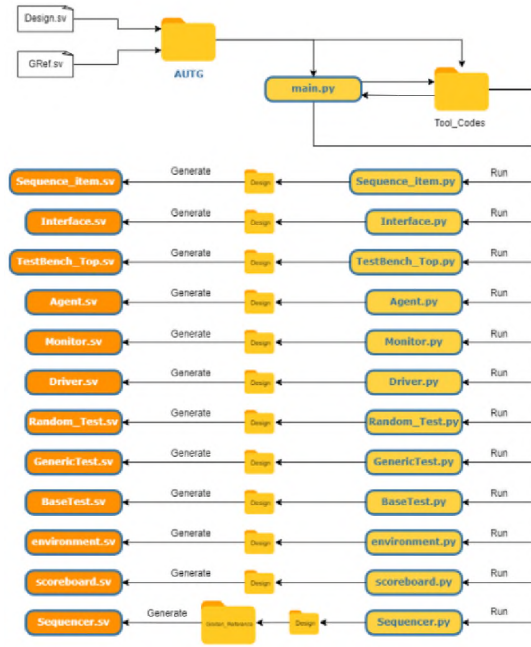

Рис. 6. Схема генерации AUTG

Применение данного метода позволяет генерировать тестовые среды для различных цифровых ИС, однако для этого требуется модель компаратора, ориентированная на конкретную методологию, на основе которой формируются тестовые векторы. Кроме того, представленная методология не позволяет создавать связи, моделирующие поведение СОЗУ, а также её управляющих блоков, включая интерфейсы AXI, AHB, APB семейства AMBA и структуры взаимодействия блоков управления СОЗУ.

Генератор СОЗУ из набора инструментов Vivado позволяет генерировать до 2-портовой ОЗУ, что является серьезным недостатком для тестирования современных систем. Кроме того, с помощью этого генератора невозможно создавать СОЗУ с управляемыми битами маски. Еще одной важной проблемой является то, что этот генератор не обеспечивает взаимодействия, поскольку сгенерированные СОЗУ

могут использоваться только в инструментарии Vivado. Кроме того, это довольно длительный процесс и требует использования других IP-адресов (рис. 7).

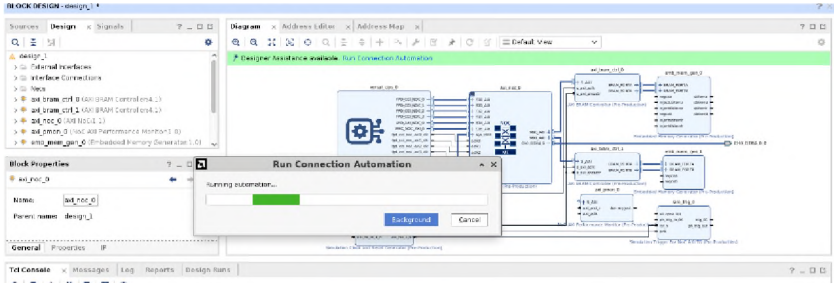

Рис. 7. Использование генератора СОЗУ в наборе инструментов Vivado

Анализ существующих методов показывает, что хотя их применение позволило частично решить перечисленные выше проблемы, они по-прежнему не удовлетворяют требованиям рынка по достижению высокой полноты проверки и допустимому времени, затрачиваемому на функциональную верификацию. **Во второй главе** представлены разработанные методы и даются решения проблем, описанных в первой главе.

Метод разработки высокопроизводительной тестовой среды для многопортовых СОЗУ. Как уже отмечалось, в современных ИС используются многопортовые СОЗУ, для функциональной проверки которых применение существующих методов нецелесообразно, поскольку они не позволяют проверить всю функциональность многопортовой памяти. В связи с этим был разработан и применён метод параллельного моделирования функциональной проверки многопортовой памяти с использованием UVM агентов системы верификации, позволяющий избежать пропусков в функциональном покрытии (рис. 8).

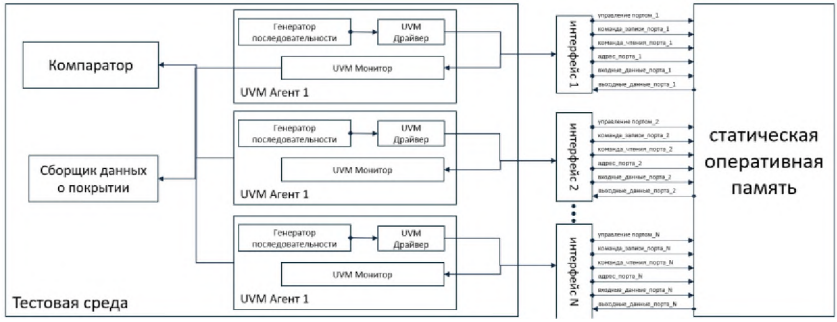

Рис. 8. Среда проверки с использованием предлагаемого метода

Применение данного метода не только сокращает время моделирования, но и увеличивает полноту функционального покрытия за счёт проверки возможных сценариев на входах в реальных рабочих режимах, включая ситуации состязаний и конфликтов между различными портами (рис. 9,10).

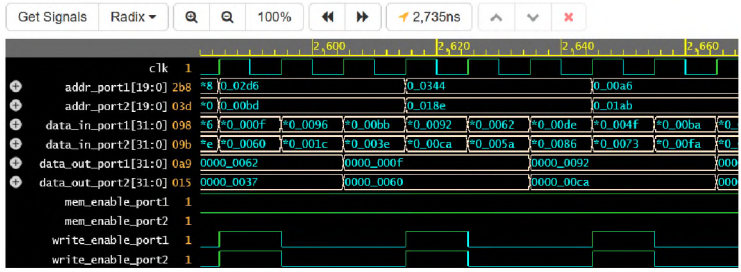

Рис. 9. Результаты моделирования двухпортового СОЗУ с использованием предлагаемого метода

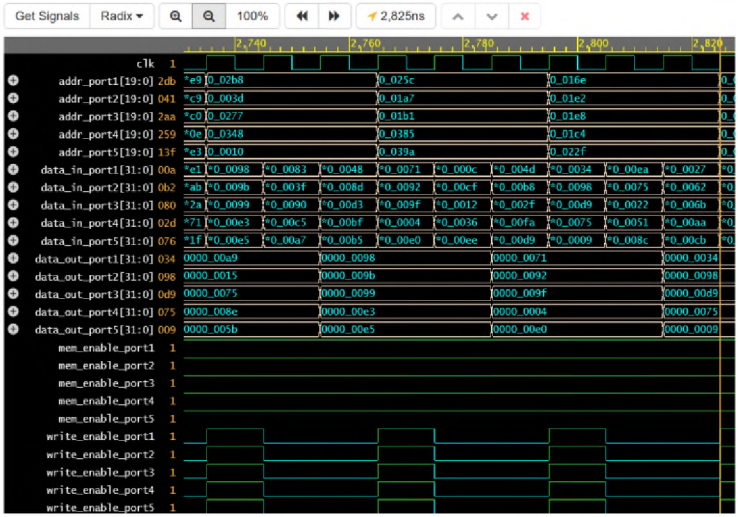

Рис. 10. Результаты моделирования пятипортового СОЗУ с использованием предлагаемого метода

Для оценки эффективности предлагаемого метода было проведено моделирование различных СОЗУ с использованием существующего и предлагаемого методов (табл. 5, рис. 11,12).

Таблица 5

Сравнение параметры проверки предлагаемого и существующего методов

Количество портов для СОЗУ	Существующий метод					Предлагаемый метод				
	2	3	4	5	6	2	3	4	5	6
Время проектирования (ч)	4	5	6	7	8	6	7,5	9	10	11
Время симуляции (ч)	32	40	49	60	71	20	24	30	37	44
Используемая память (гб)	2	2,6	3	3,3	3,8	1,5	2	2,4	2,7	3,1

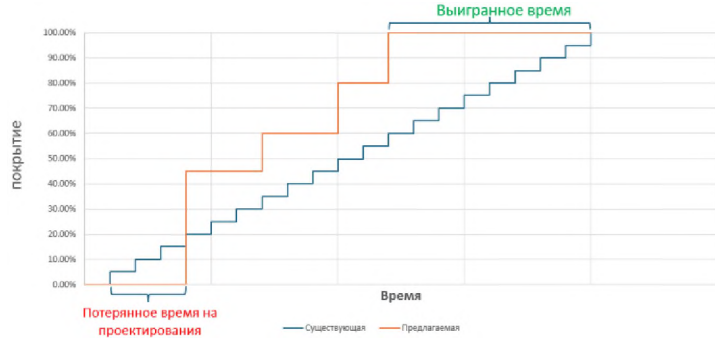

Рис. 11. Сравнение общего времени проверки с использованием предлагаемого и существующего методов

Рис. 12. Сравнение обьема используемой памяти, с применением предлагаемого и существующего методов

Таким образом, применение предлагаемого метода позволило сократить время, затраченное на моделирование, на 20%, а объём используемой памяти — на 20–30%. Однако время, затраченное на разработку, увеличилось на 50%.

Метод разработки тестовой среды для блока управления СОЗУ. Как уже было отмечено в предыдущей главе, применение современных методов верификации для функциональной проверки управляющих блоков СОЗУ с общей асинхронной и частной синхронной методологией нецелесообразно, поскольку для достижения приемлемого уровня функционального покрытия требуется значительное время на разработку и моделирование (рис. 10). В данной работе предлагается дополнить верификационную среду новым компонентом для инъекции ошибок в данные, хранящиеся в СОЗУ (рис. 13).

Рис. 13. Формирование очереди данных в элементах буферизации очереди

В предлагаемой верификационной среде в компоненте компаратора разработан механизм хранения записанных данных и команд чтения с использованием встроенных очередных массивов языка SystemVerilog. Максимальная глубина очередей соответствует глубине буферизующих элементов, используемых в интерфейсе, а количество используемых массивов соответствует числу адресов в блоке управления СОЗУ. Это позволяет сохранить равное количество данных по всем адресам. Данные считываются по следующему алгоритму:

$$read\_data_{addres(N)} = memory_{core[address][i]}, \quad if\ i+1 == READ \tag{3}$$

Для оценки эффективности предлагаемого метода был разработан блок управления СОЗУ с различной разрядностью, использующий очередные буферизующие элементы, а также блок обнаружения и коррекции ошибок на 1 и 2 бита. Были проведены различные симуляции (табл. 6 и 7, рис. 14).

Таблица 6

Результаты моделирования блока управления СОЗУ, *с использованием помехозащищенного блока с 1-битным детектированием.*

Параметры	Существующий метод			Предлагаемый метод		
Разрядность блока управления (бит)	8	16	32	8	16	32
Время проектирования (ч)	7	7	7	10	10	10
Время симуляции (ч)	2	6	8	1,5	4,5	6
Используемая память (гб)	3,2	5,5	7,6	2,7	3,9	5,6

Таблица 7

Результаты моделирования блока управления СОЗУ, *с использованием помехозащищенного блока с 2-битным детектированием.*

Параметры	Существующий метод			Предлагаемый метод		
Разрядность блока управления (бит)	8	16	32	8	16	32
Время проектирования (ч)	10	10	10	13	13	13
Время симуляции (ч)	4	12	16	2,5	9	12
Используемая память (гб)	5,5	9,1	12,4	4,3	7,2	9,4

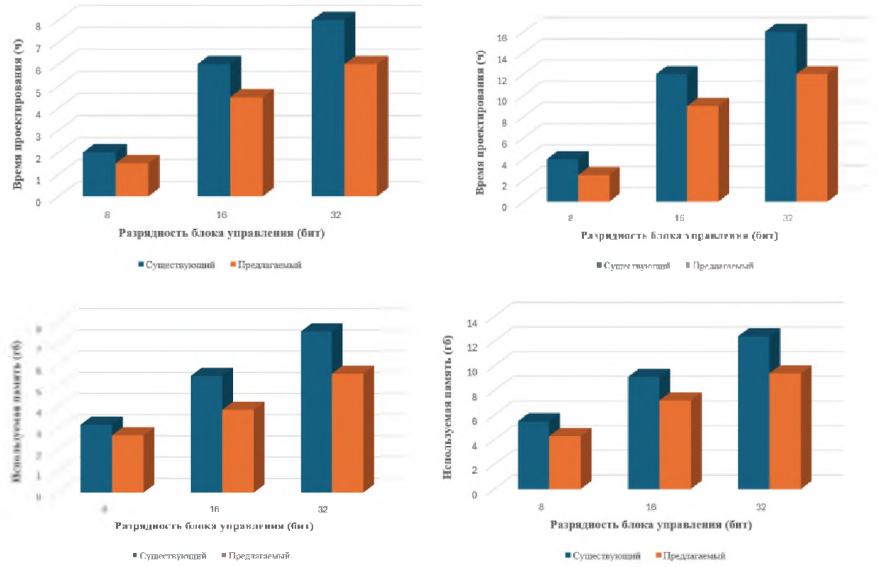

Рис. 14. Время моделирования и используемая память, с использованием предлагаемого и существующего методов

Таким образом, применение предлагаемого метода позволило сократить время, затраченное на моделирование, на 25%, а объём используемой памяти - на 20-25%. Однако время, затраченное на разработку, увеличилось на 30-45%.

Метод автоматической генерации тестовых сред для СОЗУ и их блоков управления. Проектирование быстродействующих сред тестирования для цифровых ИС требует специальных знаний в области объектно-ориентированного программирования. Разработка такой среды тестирования требует написания большого объема кода, что усложняет применение этих методов. На рынке используются автоматизированные инструменты генерации для проектирования тестовых сред, которые сокращают время, необходимое для него, однако существующие инструменты также имеют недостатки. Для создания тестовых сред используются библиотеки, описываемые шаблонами языка программирования. Использование этих шаблонов требует глубоких знаний соответствующего языка программирования, кроме того, их использование создает сложности для применения встроенных библиотек. Встроенные библиотеки потребляют дополнительную память и увеличивают время выполнения программы.

Предложен метод автоматической генерации тестовых сред для элементов памяти и их блоков управления, что позволяет упростить процесс проектирования тестовых сред (рис 15). Метод представляет новый шаблон на основе языка программирования Python, который не содержит никаких встроенных библиотек и проще в использовании, чем другие шаблоны, это делает его похожим на язык программирования Python.

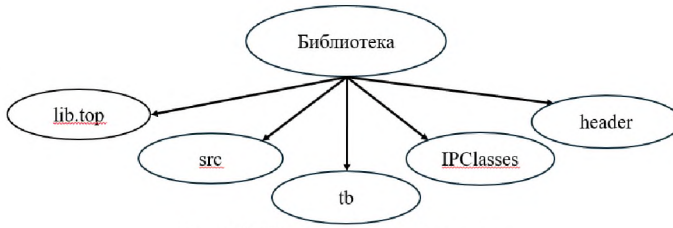

Рис. 15. Структура библиотеки

Следуя простым правилам этого шаблона, можно проектировать библиотеки, которые будут генерировать различные среды в зависимости от входных параметров. Все файлы с расширением tpn представляют собой классы языка Python, которые вызываются в методе generate для генерации соответствующего описания. Все заголовки файлов tpn должны иметь одинаковое описание (рис 16).

```
##
##  ##############################################################################################################
##
##          for PortID in {self.header.port_variables.keys()}: {>
##              self.header.get_ports_for_portID(PortID)
##              if self.header.port_type == "Read": {>
##
##                  if self.header.read_mask_for_portID: {>
    // MaskMapping for {PortID} port
    logic [{self.header.data_out_range_for_portID} read_mask_mapping_for_{PortID};
    assign read_mask_mapping_for_{PortID} = {{ {self.header.read_mask_mapping_for_portID} }};
##                  <}
##
    // Read Operation for {PortID} port
    always_ff @(posedge {self.header.clock_for_portID}) begin
        if ({self.header.memory_enable_for_portID})
##                  if self.header.read_enable_for_portID: {>
            if({self.header.read_enable_for_portID})
##                  <}
##                  if self.header.read_mask_for_portID: {>
                foreach (read_mask_mapping_for_{PortID}[mask_bit]) begin
                    if (read_mask_mapping_for_{PortID}[mask_bit])
                        {self.header.data_out_for_portID}[mask_bit] <= memory_core[{self.header.address_for_portID}][mask_bit];
                end
##                  <}
##                  else: {>
                {self.header.data_out_for_portID} <= memory_core[{self.header.address_for_portID}];
##                  <}
    end
##
##              <}
##              elif self.header.port_type == "Write": {>
##
##                  if self.header.write_mask_for_portID: {>
    // MaskMapping for {PortID} port
    logic [{self.header.data_in_range_for_portID} write_mask_mapping_for_{PortID};
    assign write_mask_mapping_for_{PortID} = {{ {self.header.write_mask_mapping_for_portID} }};
##                  <}
##
    // Write Operation for {PortID} port
    always_ff @(posedge  {self.header.clock_for_portID}) begin
        if ({self.header.memory_enable_for_portID})
##                  if self.header.write_enable_for_portID: {>
            if({self.header.write_enable_for_portID})
##                  <}
##                  if self.header.write_mask_for_portID: {>
```

Рис. 16. Пример шаблонного IP-кода tpn

Такое описание, по-видимому, делает описание кода более читабельным, поскольку сгенерированный код легко отличить от кода, описанного на Python. Использование библиотеки, описанной в данном методе, сокращает время проектирования тестовой среды на 40–70%. Еще одним преимуществом является то, что библиотеки можно легко интегрировать в другие ПС, что позволяет настраивать IP-адреса, генерировать их описания и тестовые среды. С помощью разработанных библиотек были спроектированы СОЗУ различных размеров, а также проведены различные имитационные эксперименты с использованием предлагаемого метода автоматической генерации для оценки времени проектирования и функциональной покрытии (табл. 8, рис. 17,18).

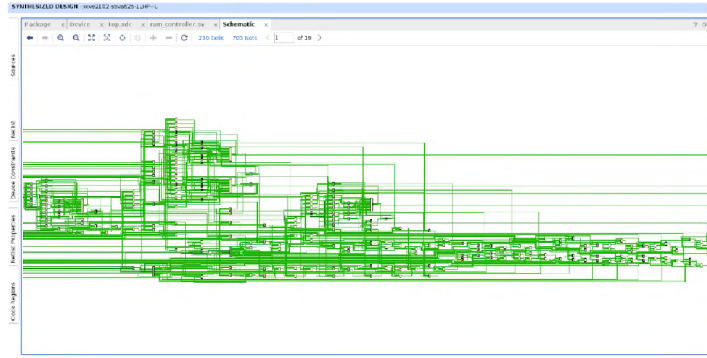

Рис. 17. Разработанные СОЗУ

Таблица 8

Время, затрачиваемое на проектирование многопортовой ЗУ и охват проверки, достигнутая с помощью предлаяаемого метода

Количество портов для СОЗУ	Существующий метод		Предлагаемый метод	
	Время проектирования (ч)	Функциональное покрытие (%)	Время проектирования (ч)	Функциональное покрытие (%)
2	5	100	0,3	100
3	7	100	0,3	100
4	8	100	0,3	100
5	9	100	0,3	100
6	10	100	0,3	100

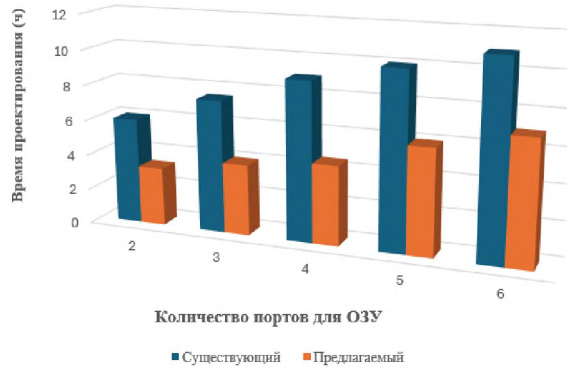

Рис. 18. Сравнение общего времени проектирования с использованием предлагаемого и существующего методов

Метод представляет новый шаблон на основе языка программирования Python, который не содержит никаких встроенных библиотек и проще в использовании, чем другие шаблоны, это делает его похожим на язык программирования Python.

В третьей главе разработано ПС “Design generator and validator” (DGV), которое сокращает время, затрачиваемое на проектирование и моделирование схем, за счет уменьшения ошибок, вызванных человеческим фактором.

Графическая среда ПС состоит из следующих компонентов: главной панели команд, вспомогательной панели команд, рабочей области и консоли ввода команд. В начале работы вспомогательная панель команд неактивна. На первом этапе необходимо создать новый проект или открыть уже сохранённый. После этого откроется среда проекта, и команды вспомогательной панели станут активными (рис. 19).

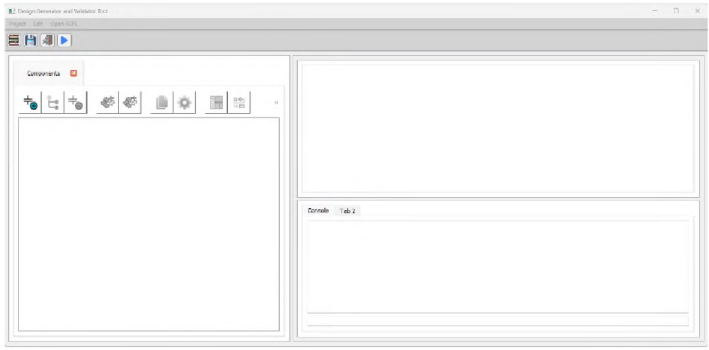

Рис. 19. Начальная графическая среда открытого проекта в ПС DGV

В рабочей области добавлен дополнительный раздел, в котором отображаются компоненты текущего проекта. Для добавления новых компонентов необходимо импортировать библиотеки ИСП, выбрав Project > Libraries на главной панели команд или нажав первую кнопку на вспомогательной панели команд (рис. 20).

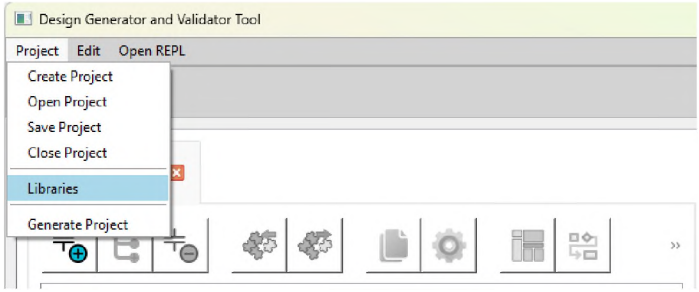

Рис. 20. Команды для добавления библиотек в открытом проекте

После импорта библиотек можно создать новый компонент ИСП, нажав на первый значок в подразделе component проекта, затем в открывшемся окне выбрать

нужную библиотеку и нажать кнопку Configure Component. После этого откроется дополнительное окно для настройки параметров компонента (рис. 21).

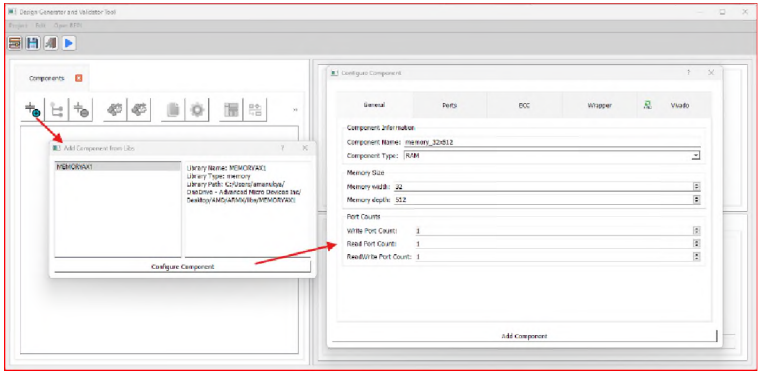

Рис. 21. Последовательность команд для добавления нового компонента в проект

После выбора компонентов активируются все команды панели управления, которые описаны в десертации (рис. 22).

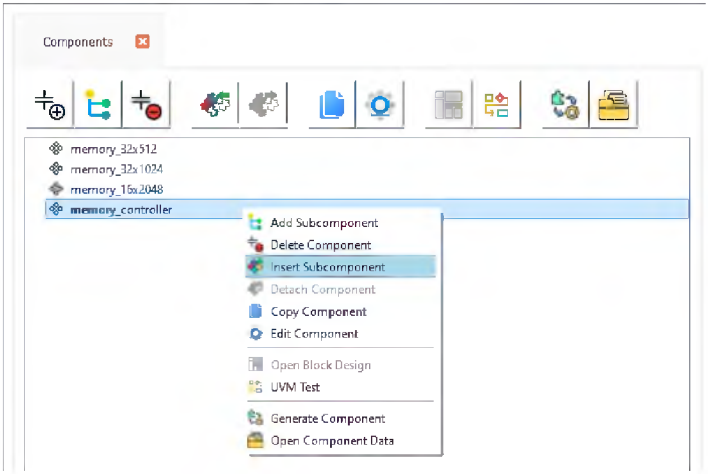

Рис.22. Список всех доступных команд для нового компонента в проекте

При нажатии кнопки “сгенерировать” указанный компонент и его подкомпоненты также будут сгенерированы. У имён сгенерированных компонентов значок окрашивается в синий цвет, что позволяет отличать их как сгенерированные.

Для проверки и обоснования эффективности разработанного ПС. С его помощью были спроектированы ИСП различных размеров и конфигураций, управляющие блоки ИСП, сгенерированы их описания на языке SystemVerilog RTL, а также другие необходимые для проектирования файлы. Проведены моделирование и симуляция с использованием инструментария Vivado компании AMD (табл. 9-11).

Таблица 9

Сравнение параметров проверки с использованием сгенерированного метода ПС DGV с существующим методом

Количество портов для СОЗУ	Существующий метод			DGV метод		
	Время проектирования (ч)	Время симуляции (ч)	Используемая память (гб)	Время проектирования (ч)	Время симуляции (ч)	Используемая память (гб)
1	3	20	1,5	0.5	15	1
2	4	32	2	0.5	20	1,5
3	5	40	2,6	0.5	24	2
4	6	49	3	0.5	30	2,4
5	7	60	3,3	0.5	37	2,7
6	8	71	3,8	0.5	44	3,1

Таблица 10

Сравнение параметров проверки блока управления СОЗУ, на основе помехозащищенного блока с 1-битным детектированием с использованием сгенерированного метода ПС DGV с существующим методом

Параметры	Существующий метод			DGV метод		
Разрядность блока управления (бит)	8	16	32	8	16	32
Время проектирования (ч)	7	7	7	2	2	2
Время симуляции (ч)	2	6	8	1,5	4,5	6
Используемая память (гб)	3,2	5,5	7,6	2,7	3,9	5,6

Таблица 11

Сравнение параметров проверки блока управления СОЗУ, на основе помехозащищенного блока с 1-битным детектированием с использованием сгенерированного метода ПС DGV с существующим методом

Параметры	Существующий метод			DGV метод		
Разрядность блока управления (бит)	8	16	32	8	16	32
Время проектирования (ч)	10	10	10	3	3	3
Время симуляции (ч)	4	12	16	2,5	9	12
Используемая память (гб)	5,5	9,1	12,4	4,3	7,2	9,4

Таким образом, из результатов становится ясно, что с помощью ПС DGV и с использованием предложенных методов проектирования быстродействующих верификационных сред удалось сократить время проектирования в 3-5 раз. Применение инструмента также исключает вероятность ошибок, связанных с человеческим фактором.

ОСНОВНЫЕ ВЫВОДЫ ПО ДИССЕРТАЦИОННОЙ РАБОТЕ

1. Предложен метод разработки быстродействующих средств верификации многопортовых интеллектуальных модулей памяти (ИМП), основанный на использовании агентов ИМП и параллельной симуляции, что позволило сократить время симуляции на 20% и объем используемой памяти на 20-30%, но увеличило время проектирования на 50%. [3].
2. Разработан метод создания быстродействующих средств верификации управляющего блока СОЗУ, который за счет применения очередей обеспечивает высокий уровень покрытия при снижении времени симуляции на 25% и объема используемой памяти на 20–35%, при этом увеличив время проектирования.при этом время проектирования увеличивается на 30-45% [4].
3. Разработан метод автоматической генерации быстродействующих сред верификации цифровых ИС на основе новых библиотек языка Python, применение которого снижает время проектирования на 40–50% [2,5].
4. Разработан ПС DGV (Design Generator and Validator) для автоматизированного проектирования средств быстродействующей верификации цифровых ИС, позволяющий создавать функциональные среды верификации на основе современных методологий с внедрением предложенных методов. Использование ПС при проектировании и верификации СОЗУ сокращает время проектирования в 5 раз, а для их управляющих блоков — примерно в 3 раза, позволяя одновременно создавать быстродействующие среды верификации для различных интеллектуальных модулей с различными конфигурациями за счёт увеличения объема используемой памяти на 15% [1-5].

Основные результаты диссертации опубликованы в следующих работах:

1. Туманян А.К., Манукян А.Г., Галстян А.А., Даниелян А.М / Аппаратные акселераторы на основе систолических матриц // Вестник НПУА: Информационные технологии, электроника, радиотехника. – 2020. – № 1. – С. 44-56.
2. Ամանկյան Ա.Գ., Գալստյան Ա.Ա., Դանիելյան Ա.Մ. Ձուգահեռ բազմակարգ բազմապատկիչների պարամետրացված մոդելի մշակում // ՀՀ ԳԱԱ և ՀԱՊՀ-ի Տեղեկագիր. Տեխն. Գիտ. սերիա. - 2020. - Հատ. 73, N 4. - Էջ 442-448:
3. Petrosyan O. and Manukyan A. Functional Verification of Multiport SRAM Memories Based on UVM // 2023 IEEE East-West Design & Test Symposium (EWDTS). - Batumi, Georgia, 2023 P. 1-4, doi: 10.1109/EWDTS59469.2023.10297116.
4. Manukyan A. Design and UVM Based Verification of FTL Memory Controller // RA NAS and NPUA. Ser. of tech. sei. – 2024 - v77 N3 – P. 367-374, doi: 10.53297/0002306X-2024.v77.3-367.
5. Petrosyan O., Manukyan A. Danielyan A., Khachatrian S. Introduction of Customizable RTL and UVM Testbench Generator // Proceedings of NPUA: Information Technologies, Electronics, Radio Engineering - 2024 - № 2. – P. 86-96, doi: 10.53297/18293336-2024.2-86.

ԱՄՓՈՓԱԳԻՐ

Ժամանակակից ինտեգրալ սխեմաների ֆունկցիոնալության բարդության աճի հետ մեկտեղ ավելանում է ինտեգրալ սխեմաներում ֆունկցիոնալ թերությունների հավանականությունը: Այդ իսկ պատճառով ինտեգրալ սխեմաների նախագծման յուրաքանչյուր փուլից հետո իրականացվում է ֆունկցիոնալ ստուգում, որը կազմում է ընդհանուր նախագծի վրա ծախսված ժամանակի մոտ 80 %-ը:

Ինտեգրալ սխեմաների բարձր արագագործությամբ ստուգման միջավայրերի նախագծման առկա մեթոդները չեն բավարարում շուկայում պահանջվող ստուգման բարձր ծածկույթի և ֆունկցիոնալ ստուգման վրա ծախսված ժամանակի պահանջներին: Ուստի նոր մոտեցումների մշակումը դարձել է անհրաժեշտություն:

Ատենախոսությունը նվիրված է թվային ինտեգրալ սխեմաների արագագործ ստուգման միջոցների արդի խնդիրների լուծմանը:

Առաջարկվել է բազմապորտ ստատիկ օպերատիվ հիշող սարքերի (ՍՕՀՍ) արագագործ ստուգման միջոցների մշակման մեթոդ հիմնված ստուգման համապիտանի մեթոդաբանությամբ աշխատող ագենտների կիրառման և զուգահեռ նմանրկման վրա, ինչը թույլ է տվել կրճատել նմանակման վրա ծախսված ժամանակը 20%-ով իսկ նմանրկման համար օգտագործված հիշողությունը 20-30%-ով, սակայն նախագծման վրա ծախսված ժամանակը ավելացել է 50%-ով:

Ստեղծվել է ՀԲՏ-ով աշխատող ՍՕՀՍ-ի ղեկավարման բլոկի բարձր արագագործությամբ ստուգման միջոցների մշակման մեթոդ, որը հերթային զանգվածների կիրառման շնորհիվ ապահովում է ստուգման բարձր ծածկույթ կրճատելով նմանակման վրա ծախսված ժամանակը 25%-ով իսկ նմանրկման համար օգտագործված հիշողությունը 20-35%-ով, սակայն նախագծման վրա ծախսված ժամանակը ավելացել է 30-45%-ով:

Մշակվել է թվայի ինտեգրալ սխեմաների բարձր արագագործությամբ ստուգման միջավայրերի ավտոմատ գեներացման մեթոդ python լեզվի նոր գրադարանների նախագծման հիման վրա, որի կիրառումը նվազեցնում է նախագծման վրա ծախսված ժամանակը 40-50%-ով:

Մշակվել է թվային ինտեգրալ սխեմաների բարձր արագագործությամբ ստուգման միջոցների ավտոմատ նախագծման Design Generator and Validator (DGV) ծրագրային գործիքը, որը թույլ է տալիս ստեղծել թվային ինտեգրալ սխեմաների ֆունկցիոնալ ստուգման միջավայրեր՝ արդի մեթոդաբանությունների հիման վրա: Այն ներդրվել է «Զայլինքս Արմենիա» ՍՊԸ-ում: Մշակված Design Generator and Validator (DGV) ծրագրային գործիքի կիրառումը զգալիորեն հեշտացրել է թվային սխեմաների ստուգման գործընթացը, և կրճատել դրանց վրա ծախսվող ժամանակը 3-5 անգամ:

ARMAN GEVORG MANUKYAN

DEVELOPMENT OF HIGH-SPEED TESTING TOOLS FOR DIGITAL INTEGRAL CIRCUITS

SUMMARY

With the increasing complexity of the functionality of modern integrated circuits (ICs), the likelihood of functional errors in these circuits also increases. Therefore, functional verification is carried out after each stage of IC design, accounting for approximately 80% of the total project time. As a result, the development of high-speed verification environments for ICs has become one of the most pressing challenges.

Since memory intellectual properties (IPs) constitute up to 70% of the overall design in modern computing systems, their correct operation is critically important. In modern ICs, memory IPs are not isolated units but are tightly integrated with other functional blocks. Logic blocks, including combinational and sequential elements, often require memory to temporarily store the states of finite state machines.

The existing methods for designing high-speed verification environments for ICs do not meet the market demands for high verification coverage and acceptable time spent on functional testing. Therefore, the development of new approaches has become a necessity.

This dissertation is dedicated to solving current problems in high-speed verification of digital integrated circuits.

A method has been proposed for developing high-speed verification tools for multi-port static random-access memory (SRAM), based on the use of agents operating under a universal verification methodology and parallel simulation. This approach has reduced simulation time by 20% and memory usage during simulation by 20–30%, although the design time increased by 50%.

A method has been developed for creating high-speed verification tools for an SRAM controller operating with a finite state machine (FSM). By using queue-based arrays, the method ensures high verification coverage while reducing simulation time by 25% and memory usage by 20–35%. However, the design time increased by 30–45%.

A method for automatic generation of high-speed verification environments for digital integrated circuits has been developed, based on the creation of new Python libraries. The use of this method reduces design time by 40–50%.

An automated tool for designing high-speed verification environments for digital integrated circuits, named Design Generator and Validator (DGV), has been developed. It allows for the creation of functional verification environments for digital integrated circuits based on modern methodologies. The tool has been adopted by Xilinx Armenia LLC. The use of the developed DGV tool has significantly simplified the verification process for digital circuits and reduced the required verification time by 3–5 times.