ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ ԿՐԹՈՒԹՅԱՆ, ԳԻՏՈՒԹՅԱՆ,
ՄՇԱԿՈՒՅԹԻ ԵՎ ՍՊՈՐՏԻ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

Հակոբյան Հովհաննես Համլետի

ԷԼԵԿՏՐՈՆԱՅԻՆ ՍԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ ՆԱԽԱԳԾՄԱՆ ՀԱՄԱԿԱՐԳԵՐԻ ՌԵԳՐԵՍԻՈՆ ԹԵՍՏԱՎՈՐՄԱՆ ՄԻՋՈՑՆԵՐԻ ՄՇԱԿՈՒՄԸ

ՍԵՂՄԱԳԻՐ

Ե.13.02 – «Ավտոմատացման համակարգեր» մասնագիտությամբ տեխնիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսության

Երևան 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ , НАУКИ, КУЛЬТУРЫ И СПОРТА
РЕСПУБЛИКИ АРМЕНИЯ

НАЦИОНАЛЬНЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ АРМЕНИИ

Акобян Оганнес Гамлетович

РАЗРАБОТКА СРЕДСТВ РЕГРЕССИОННОГО ТЕСТИРОВАНИЯ СИСТЕМ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ ЭЛЕКТРОННЫХ СХЕМ

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата технических наук
по специальности 05.13.02– “Системы автоматизации”

Ереван 2020

Ատենախոսության թեման հաստատվել է Հայաստանի ազգային պոլիտեխնիկական համալսարանում

Գիտական ղեկավար՝ տ.գ.դ. Ա.Գ. Հարությունյան

Պաշտոնական ընդդիմախոսներ՝ տ. գ. դ. Հ. Ս. Սուքիասյան
տ. գ. թ. Հ. Գ. Կասարջյան

Առաջատար կազմակերպություն՝ ՀՀ ԳԱԱ ինֆորմատիկայի և ավտոմատացման պրոբլեմների ինստիտուտ

Ատենախոսության պաշտպանությունը տեղի կունենա 2020թ. օգոստոսի 20-ին, ժամը 14:00-ին, ՀԱՊՀ-ում գործող - «Կառավարման և ավտոմատացման» 032 Մասնագիտական խորհրդի նիստում (հասցեն՝ 0009, Երևան, Տերյան փ., 105, 17 մասնաշենք)
Ատենախոսությանը կարելի է ծանոթանալ ՀԱՊՀ–ի գրադարանում:
Սեղմագիրն առաքված է 2020 թ. հունիսի 19-ին:

032 Մասնագիտական խորհրդի

գիտական քարտուղար, տ.գ.թ.՝ Ա.Վ. Մելիքյան

Тема диссертации утверждена в Национальном политехническом университете Армении (НПУА)

Научный руководитель: д.т.н. А.Г. Арутюнян
Официальные оппоненты: д.т.н. А.С. Сукиасян
к.т.н. О. Г.Касарджян

Ведущая организация: Институт проблем информатики и автоматизации НАН РА

Защита диссертации состоится 20 августа 2020г. в 14:00 ч. на заседании Специализированного совета 032 - “Управления и автоматизации”, действующего при Национальном политехническом университете Армении (НПУА), по адресу: 0009, г. Ереван, ул. Теряна, 105, корпус 17.
С диссертацией можно ознакомиться в библиотеке НПУА.
Автореферат разослан 19 июня 2020г.

Ученый секретарь

Специализированного совета 032 к.т.н. А.В. Меликян

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы. Развитие микроэлектронных технологий приводит к стремительному росту интеграции интегральных схем (ИС), что диктует необходимость создания устранять с большими функциональными возможностями и меньшими размерами. Одной из основных задач в области развития и усовершенствования микроэлектронных технологий является создание микроминиатюрных устройств с высокой воспроизводимостью и функциональной сложностью. Такие схемы используются в высокопроизводительных цифровых устройствах, интеллектуальных устройствах (телефоны, часы и т.д.), медицинских приборах в других областях. Развитие интеграции ИС характеризуется законом Мура, согласно которому число транзисторов удваивается каждые 18 месяцев. В условиях ограниченной площади поверхности полупроводниковой платформы с уменьшением размеров элементов количество элементов в единице объема ИС увеличивается, что приводит к усложнению ИС, и, как следствие к неуклонному повышению сложности процесса их проектирования.

В результате усложнения ИС возрастают затраты на их разработку и следовательно на систем автоматизированного проектирования (САПР) электронных схем.

Это связано прежде всего с затратыами на программное обеспечение (ПО) САПР электронных схем, поскольку параллельно с усложнением схем создаются современные программные средства, повышающие эффективность проектных работ. В то же время параллельно с улучшением ПО разрабатываются и совершенствуются средства тестирования проекта ИС. Роль последнего еще более важна с точки зрения обеспечения стабильности и надежности проекта в современных ИС.

В условиях динамичного развития современных микроэлектронных устройств возникает необходимость постоянного усовершенствования средств их тестирования. В результате этого, по мере возрастания функциональности ИС, растет и объем регрессионного тестирования в средствах их тестирования. Регрессионное тестирование проводится после внесения изменений в проекте с целью проверки работоспособности новой версии проекта. В настоящее время существует четыре известных способа тестирования проектов в области регрессионного тестирования. Первый способ основан на целостном тестировании проекта, т.е., независимо от количества изменений, внесенных в проект, и значимости введенных в проектирование изменений, перед выпуском новой версии вводится в действие полный пакет регрессионных тестов. Недостаток этого способа заключается в том, что увеличиваются время и затраты тестирования, так как проектируемые схемы периодически усложняются, а количество тестовых пакетов и случаев увеличивается.

Второй способ основан на выборочном тестировании. Существует несколько вариантов отбора тестов: произвольный, целенаправленный и отбор тестов вручную. Для применения этого способа нужно быть уверенным, что система полностью протестирована в результате выбранных тестов, и не осталось нетестированных частей.

Третий способ тестирования основан на классификации тестов. Классификация производится в соответствии с означимостью тест-кейсов, т.е. в соответствии со степенью сложности функциональности, проверяемой в результате работы тестов. Этот способ также имеет недостаток, так как, хотя тесты классифицируются и используются в соответствии с их значимостью, возможны случаи, когда пропуск менее важных тестов может привести наличию в проекте нетестированных частей.

Четвертый способ тестирования называется гибридныым и основан на комбинации второго и третьего сспособов. В этой версии отбор тестов и классификация осуществляются одновременно и характеризуются более или менее одними и теми же недостатками.

Обобщая вышесказанное, можно констатировать, что, несмотря на некоторые успехи в решении указанной проблемы, вследствие увеличения сложности проектов и развития их функциональности внедрение новых средств регрессионного тестирования в электронных системах автоматизации проектирования и их постоянное совершенствование становятся актуальными задачами. В частности, наряду с усложнением процесса проектирования актуальным становится вопрос обеспечения стабильности работы проектируемого объекта. Другими словами, добавление необходимых новых функциональных объемов и исправление ошибок не должны отрицательно влиять на работоспособность текущего проекта.

Наряду с развитием регрессионного тестирования возникает ряд новых проблем, наиболее важными из которых являются увеличение объема тестовых пакетов, необходимых для тестирования, и в связи с этим увеличение времени тестирования. Значимость последних больше подчеркивается в таких процессах проектирования, которые выполняются с помощью краткосрочных выпусков.

Для решения вышеперечисленных задач и с целью устранения имеющихся недостатков тестирования САПР электронных схем, предлагаются механизмы отбора регрессионных тестов, основанные на алгоритмах хэширования. Для целей отбора тестов вместо матрицы прослеживаемости целесообразно использовать разработанную новую структуру, которая основывается на сохранении данных в формате JSON.

Процесс классификации тестов в механизме гибридного отбора тестов предполагается проводить с учётом критических степеней, используя при этом структурный декоратор-шаблон проектирования.

Исходя из вышесказанного, приходим выводу, что в настоящее время разработка средств сокращения сроков регрессионного тестирования, которые непосредственно зависят от изменений объемов функциональности разрабатываемого продукта, при этом одновременно обеспечивая бесперебойную работу версий проекта, является актуальной задачей.

Предмет исследования. Средства методического, алгоритмического и программного обеспечения отбора регрессионных тестов и тестирования ИС в САПР электронных схем, с учетом усложнения функциональности текстового описания представления ИС, расширения и изменения проекта.

Цель работы. Разработка и исследование эффективных методов, алгоритмических и программных средств регрессионного тестирования САПР электронных схем, в условиях изменения функциональности проектов ИС.

Методы исследования. В процессе работы над диссертацией использованы файлы текстового описания представления ИС, с помощью которых описывается функциональность запоминающих устройств. В работе также использованы несколько видов алгоритмов хэширования, а в случаях с большим объемом данных системы быстрого поиска, продуктивные методы организации сохранения данных, а также элементы QT библиотеки ПО САПР, некоторое количество структурных и поведенческих программных шаблонов.

Научная новизна

1. Предложены и разработаны методы отбора регрессионных тестов ИС в САПР электронных схем, позволяющие избежать полного регрессионного тестирования.
2. Предложен метод выявления функциональных изменений в проекте на основе алгоритмов хэширования. Для сокращения времени, затрачиваемого для обнаружения проектных изменений в САПР электронных схем, в результате исследований выбран наиболее быстродействующий алгоритм хэширования.
3. Предложена новая структура хранения данных на основе формата JSON, которая обеспечивает эффективность в САПР электронных схем в контексте увеличения объема тестовых пакетов.
4. Предложена поисковая система на основе фильтрации Блума в механизме отбора регрессионных тестов с целью сокращения времени, затрачиваемого на поиск тестов в крупномасштабных данных.
5. Предложен механизм классификации, основанный на критических степенях, который обеспечивает двойную фильтрацию тест-кейсов в избранных тестовых пакетах. В основу разработанного механизма заложен программный структурный шаблон декоратор.

Практическая ценность работы. Методы, предложенные в диссертации, и полученные результаты были реализованы в САПР элекетронных схем, при проектировании ИС с учетом изменения их функциональности. Разработаны

механизмы и программное средство отбора регрессионных тестов для тестирования ИС. Последнее позволяет получить список измененных файлов для следующей версии проекта и выдает в качестве выходного результата тестовые пакеты, которые необходимо запустить для проверки эффекта от внесенных изменений. Благодаря разработанного программному обеспечению стало возможным отделить от выбранного тестового пакета наиболее важные тесты, т.е тесты имеющие высокую критическую степень. Этот двойной механизм фильтрации был реализован с использованием структурного декоратора-шаблона. ПО разработано на языках программирования Python 3, Qt 5 и в соответствующих средах, предназначенных для использования в операционных системах Linux и Windows.

Использование созданного программного средства для тестирования реальных проектов показало его высокую эффективность. Оно может быть интегрировано в существующие САПР в качестве подсистемы автоматизированного регрессионного тестирования.

Достоверность научных положений подтверждается данными тестирования проектов в ряде тестовых схем, созданных на основе предложенных методов, а также программным инструментарием для отбора регрессионных тестов в условиях изменения функциональности ИС.

Внедрение. Разработанное программное средство регрессионного тестирования внедрено в ЗАО “Синопсис Армения” и используется в САПР электронных схем для проверки функциональности текстового описания запоминающих устройств.

Основные положения, выносимые на защиту.

- автоматизированное средство выявления измененных участков в условиях изменения функциональности ИС;
- способ эффективного хранения данных, устанавливающий связь между тестами и функциями.;
- гибридный механизм отбора и классификации тестов, основанный на структурном шаблоне;
- эффективный способ поиска тестов в условиях большого объема данных.

Апробация работы.

- Международный симпозиум по проектированию и тестированию "EWDTS: IEEE East-West Design & Test Symposium", (Батуми, Грузия, 2019) ;
- Научных семинарах кафедры “Микроэлектронных схем и систем” НПУА (г. Ереван, Армения, 2017-2019).

Публикации. Основные положения диссертации отражены в пяти научных публикациях, две из которых включены в базу Scopus, а две публикации-без

соавторов. Еще одна статья в настоящее время находится в печати. Список публиикаций приведен в конце автореферата.

Структура и объем диссертации. Диссертация состоит из введения, четирех глав, списка литературы из 101 наименовани и четирех приложений. Основной текст диссертации составляет 114 страниц, включает 47 рисунков и 12 таблиц, диссертация написана на армянском языке.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении обоснована актуальность темы, сформированный, цель и задачи работы, представлены методы исследования, научная новизна, практическая ценность работы,а также основное положения, выносимые на защиту.

В первой главе проанализировано влияние увеличения степени интеграции ИС на сложность их проектирования. В условиях ограниченности площади полупроводниковой платформы и уменьшения размеров элементов их количество в единице объема ИС соответственно увеличивается, что приводит к усложнению самих интегральных схем. Учитывая вышесказанное, можно прийти к выводу, что параллельно с увеличением количества элементов в ИС возрастает сложность процесса их проектирования. Проект ИС представляет собой группу файлов, с помощью которах однозначно можно создавать и тестировать проектируемые ИС (рис.1).

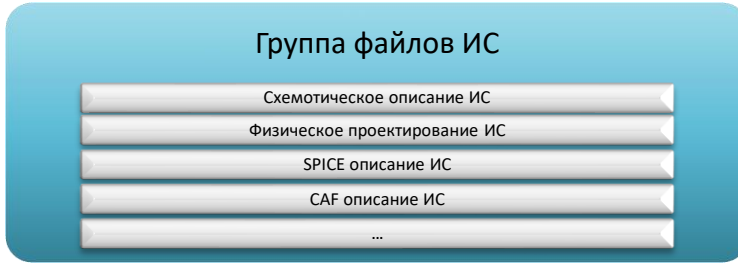

Рис.1. Содержание проекта ИС

Для проектирования современных сложных ИС необходимы такие системы ПО, с помощью которых будет возможно:

- существенно сократить сроки проектирования;
- использовать современные библиотеки;
- снизить вероятность возникновения ошибок;
- обеспечить стабильность работы ПО

Для достижения наилучшего результата жизненный цикл ПО осуществляется поэтапно в предусмотренные периоды (рис.2).

Рис.2. Жизненный цикл программного обеспечения

Для обеспечения систем ПО с указанными выше качествомы современные программные средства САПР электронных схем проходят по определенному жизненному циклу развития ПО, который предусмотрен для разработки, тестирования и обслуживания программных средств.

После завершения разработки проекта проводится тестирование. В процессе перехода от текущей версии проекта к его последующей версии проводится регрессионное тестирование (прогон регрессионных тестов). С целью облегчения организации процесса регрессионного тестирования для описания схем используется CAF описание схемы. CAF (Compiler Architecture File) – “язык” высокого уровня текстового описания схемы, который основывается на современных стандартах и языке программирования. Последний основывается на языке Python и ряде схемотехнических библиотек. Это означает, что данное описание является гибким, поскольку оно:

- независимо от среды создания (операционной системы);
- совместимо с рядом библиотек;
- может быть разработано в качестве модели жизненного цикла ПО;
- имеет уже готовую среду (ареал) тестирования.

Для апробации/запуска данного описания необходимо использовать уже разработанный CAF сортировочный программный инструмент, который также основывается на языке Python.

Параллельно с ростом объема тестов увеличивается продолжительность времени их работы, и поэтому важнейшей задачей является сокращение общего времени регрессионного тестирования, обеспечивая при этом высокое качество. Другими словами, проблема состоит в сокращении времени работы постоянно работающей тестовой комплектации в условиях изменения (развития)

функциональности проектов, ограниченных ресурсов, с сохранением стабильности проектируемого объекта, а также в обеспечении необходимого покрытия тестируемого проекта.

Вторая глава посвящена разработке методов рессионного тестирования САПР электронных схем. Известны 4 основных способа осуществления регрессионного тестирования: целостное тестирование т.е тестирование в полном объеме, отбор тестов, классификация тестов, гибридное тестирование. Гибридный вариант не что иное, как совмещенный способ “отбора тестов” и “классификации тестов”. Преимущество гибридного способа состоит в том, что в случае применения последнего отбирается и классифицируется определенная подгруппа тестов, которые соотносятся лишь с проведенными изменениями [1-3].

Разработанный механизм отбора модульных регрессионных тестов позволяет сократить количество используемых ресурсов, а также снизить общее время тестирования. С этой целью отбор тестов проводится на основе алгоритма хэширования. В качестве примера взята схема описания компилятора памяти (рис.3).

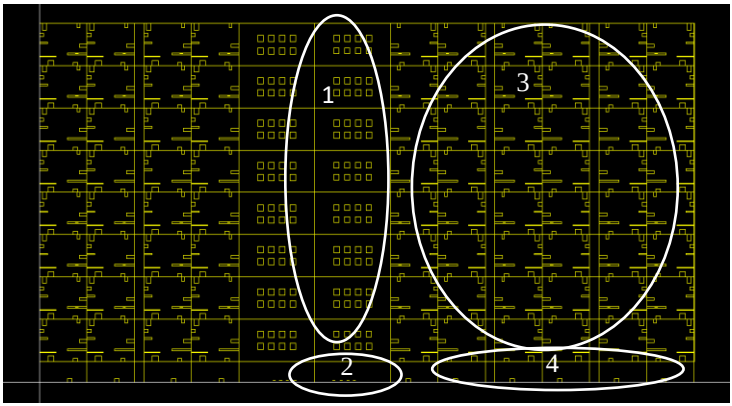

Рис.3. Компилятор памяти
1- декодер, 2- входы управления, 3- массив битовых ячеек, 4- входы/выходы данных

Ниже приведена обеспечивающая работу запоминающего устройства функциональность в CAF текстовом описании [1-3]. В правой части приведенного описания выделены произведенные изменения, в результате которых появляется необходимость регрессионного тестирования до выпуска нового варианта проекта.

```
@caf.Block
  def BuildBitRow(self, p_column,
strap_period):
      if self.cfg.get('wrap_bitcells'):
        bit_row_pattern = [self.my_bitcell,
self.my_bit2] * (strap_period / 2) +
[[self.my_strap]]
      else:
        bit_row_pattern = [self.bitcell,
self.bit2] * (strap_period / 2) +
[[self.my_strap]]

      return caf.Row(bit_row_pattern,
length=p_column)
      @caf.Block
  def BuildBitArr(self, pr, p_column,
strap_period):
        bit_row =
self.BuildBitRow(p_column=p_column,
strap_period=strap_period)

      return caf.Column([bit_row],
length=pr,
                opts=caf.Suppress('wl*'))
```

```
@caf.Block
  def BuildBitRow(self, p_column,
strap_period):
      if self.cfg.get('wrap_bitcells'):
        bit_row_pattern =
[self.another_bitcell, self.my_bit2] *
(strap_period / 2) + [[self.my_strap]]
      else:
        bit_row_pattern = [self.bitcell,
self.bit2] * (strap_period / 2) +
[[self.another_strap]]

      return caf.Row(bit_row_pattern,
length=p_column)
      @caf.Block
  def BuildBitArr(self, pr, p_column,
strap_period):
        bit_row =
self.BuildBitRow(p_column=p_column,
strap_period=strap_period)

      return caf.Column([bit_row],
length=pr,
                opts=caf.Suppress('hl*'))
```

Для решения этой задачи был применен гибридный метод отбора тестов, который позволяет производить отбор подгруппы тестов [1-4], затем из выбранной подгруппы отделять те тесты, которые имеют более высокую степень значимости. Таким образом, становится возможным повторно профильтровать отобранные тесты и сократить время целостного тестирования. На рис.4 представлена блок-схема, описывающая работу механизма отбора регрессионных тесов.

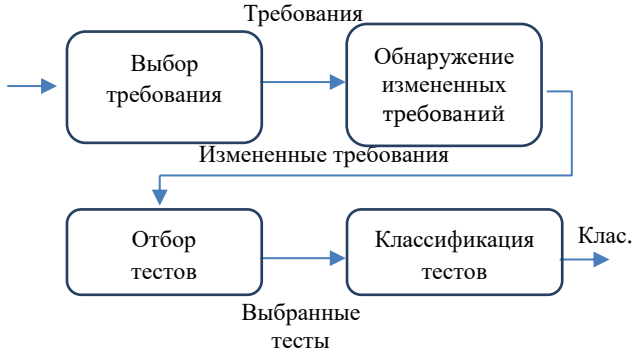

Рис. 4. Разработанная блок-схема алгоритма отбора регрессионных тестов

В блок-схеме представлены входные и выходные данные каждого блока. Блок “Выбор требований” в качестве вводных данных получает разработанный файл для

новой версии программного продукта. Получив файл, этот блок в результате находит все требования. Блок “Обнаружение измененных требований” в качестве вводных данных считывает список требований, полученных из первого блока. Затем представленные в каждом из требований команды хешируются. Для хэширования был использован алгоритм MD5 [2, 3].

Полученные хэш-значения сравниваются с исходными хэш значениями. Если хэш-значения разные, значит, в данном требовании были проведены изменения, и соответствующие последнему тесты нужно прогонять для последующей версии.

Блок “Отбор тестов” получает список измененных требований и, используя матрицу прослеживания, находит необходимые тесты для данного изменения. Последний блок “Классификация тестов” классифицирует полученный список тестов по критическим степеням.

Для более полного восприятия работы механизма отбора регрессионных тестов ниже представлен псевдокод отбора тестов.

```
Шаг 1:
foreach r∈ Requirements
foreach command_line∈ r
    hash_value = hash_value + hash(command_line)
  endfor
pair_list.append(r,hash_value)
Шаг 2:
foreach pair ∈ pair_list
  diff = compare(pair[hash_value], current_hash)
  if (diff)
    change_list.append(pair)
  endif
endfor
Шаг 3:
foreach change ∈ change_list
  change_tests = find_tests(change)
  must_run.append(chang_tests)
endfor
Шаг 4:
prioritize_tests(must_run)
```

В настоящее время в условиях постоянного увеличения объемов регрессионного тестирования ресурсы, необходимые для реализации тестирования новых объемов, могут достичь практически половины суммарных затрат, выделенных на программное обеспечение. В этих условиях задача сокращения ресурсов, необходимых для регрессионного тестирования, становится актуальной и важной.

С этой целью ниже представлен механизм повышения продуктивности регрессионного тестирования программных систем, который основан на хэшировании отбора регрессионных тестов и применении фильтров Блума. Задача повышения продуктивности тестирования современных систем автоматизации приводит к разработке таких механизмов регрессионного тестирования, которые дают возможность управлять взаимосвязью между вероятностью выявленных

ошибок в процессе тестирования и требуемыми вычислительнымих ресурсами. Для достижения вышеуказанных целей предложен метод регрессионного тестирования, основанный на хэшировании с применением фильтра Блума, при этом значения счетчиков должны различаться минимально. Применение хэширования позволяет пользоваться простыми структурами анализа и сохранения кода программы, а ввод фильтра Блума повышает скорость поиска информации. С целью установления связи между частыми требованиями в алгоритмах отбора регрессионных тестов и тест-кейсов используется матрица прослеживаемости, в которой представлена степень значимости различных тест-кейсов. Однако применение этого подхода в случае обработки большого объема данных приводит к замедлению поиска данных, поскольку с увеличением количества тестов в матрице прослеживаемости параллельно увеличиваются новые элементы, заполненные 0. Это означает, что данный тест-кейс не ответственен за требование, записанное в соответствующей строке. Таким образом, вследствие сохранения дополнительных данных применение этого метода в случаях с большим объемом данных неэффективно (см. таб. 1).

Таблица 1

Матрица прослеживаемости

Требования	*Тест-кейс*			
	Test 1	*Test 2*	…	*Test N*
createRow	3	0	…	0
createColumn	0	4	…	0
…	…	…	…	…
createTop	0	0	…	1

Для решение этой проблемы разработана новая структура сохранения данных, которая представлена в формате программных единиц JSON [5]. В предложенной структуре новые требования и их хэш-значения содержаться в соответствующем фильтре Блума, а тест – кейс добавляется в формате JSON только в строке данной функции. Это позволяет сократить хранящееся в матрице прослеживаемости количество дополнительных 0, не включая дополнительных сведений об остальных требованиях. Таким образом, мы получаем продуктивное использование памяти.

Основные преимущества, созданные на основании фильтра Блума алгоритмов, заключаются в их высокой скорости. Однако этот метод имеет также и недостатки, что обусловлено характером его вероятности, по причине которой могут встречаться “ложно положительные” результаты. “Ложно положительный” результат – это тот случай, когда получен ответ, что данный элемент существует, однако в реальности в списке данных его не существует. Отметим также, что это исключает “ложно отрицательные” результаты. “Ложно отрицательный” результат – это тот случай, когда получен ответ, что данный элемент не существует, однако в реальности в списке данных он существует.

В фильтре Блума возможно управлять вероятностью “ложно положительных” результатов, изменив величину массы бита и количество хэш-функций, благодаря чему данный механизм нашел широкое применение в случаях поиска большого объема данных.

Ниже приведены некоторые свойства фильтров Блума, которые важны с точки зрения применения механизмов отбора регрессионных тестов.

1. В отличие от фиксированной протяженности хэш-таблицы, фильтр Блума может сохранять список, который состоит из произвольного количества элементов.

2. В фильтре Блума исключена регенерация “ложно отрицательного” результата, т.е если элемент не существует в списке, то и алгоритм не покажет его наличия.

3. Удаление элементов не является возможным, так как сброс одного элемента предполагает и удаление битов, что, в свою очередь, может повлиять на результаты оставшихся элементов. Из приведенных свойств самым характерным из них является то, благодаря которому фильтр Блума никогда не выдает “ложно отрицательного” значения, что и является основной положительной стороной этого алгоритма. Важным качеством алгоритма Блума является также гибкость, т.е. возможность управления вероятностью появления “ложно положительных” результатов. Для сокращения “ложно положительных” результатов можно изменить величину массы бита, благодаря чему станет возможным избежать многократно повторяющихся битов. Возможно также увеличить количество хэш-функций, с помощью которых рассчитываются индексы массы, что также способствует сокращению “ложно положительных” результатов. Предлагаемая структура сохранения данных, представленная в формате JSON, следующая:

```
<file path1>: {
      "def_filter": <bloom filter for defs>,
      "hash_filter": <bloom filter for hashes>,
      "defs": {
<function name1>: <test path1>,
<function name2>: <test path2>
                        ….
<function nameN>: <test pathN>
      }
   },
   …
<file path N>: {
      "def_filter": <bloom filter for defs>,
      "hash_filter": <bloom filter for hashes>,
      "defs": {
<function name1>: <test path1>,
<function name2>: <test path2>
                        ….
<function nameN>: <test pathN>
      }
}
```

В вышеприведенном формате программной системы <file path1>,…, <file path N> *.py файлы с расширением, <bloom filter for defs>, <bloom filter for hashes> - фильтры Блума, созданные для сохранения требований, хэш-значений и поиска данных, <function name1>,…, <function nameN> - наименования требований в данном файле с соответствующими тестами. Ниже представлен сокращенный псевдо код разработанного механизма отбора регрессионных тестов в формате JSON [5]. Поскольку процесс отбора тестов связан с проведенными в проекте изменениями, помимо хэширования, было осуществлено выявление изменений, основанное на строчном алгоритме сравнения.

Для строчного сравнения необходимы исходные и измененные версии файлов, описывающих сам проект. С целью отделения объявления функций и формирования строчных значений, для обнаружения изменений сканируются обе версии файлов. Полученные строчные выражения сравниваются между собой с помощью алгоритмов строчного сравнения. В качестве алгоритма сравнения выбран алгоритм Хемминга [6].

Сравнивая разработанные два метода, можно сделать вывод, что механизм регрессионного тестирования, основанный на сравнении, работает в 11÷37 раз быстрее, чем алгоритм регрессионного теста, основанный на строчного сравнении. **Глава Третья** посвящена вопросам разработки алгоритмического обеспечения автоматизированных средств тестирования с учетом развития их функциональности. В общем случае отбор регрессионных тестов производится по 4-ем основным этапам: “анализ классов и функций”, “обнаружение измененных функций”, “отбор тест-кейсов”, “классификация тест-кейсов”. Осуществление этих процессов проведено методом семантического дерева. С помощью “семантического дерева” получены общие описания, определенные в каждом измененном файле проекта, классов и функций, каковыми являются: информация о теле функции, информация об аргументировании и прочее. Имея относящуюся к функциям информацию, отделяем определение функции и тело, которое с целью расчета хэш значений, в качестве вводного значения будет передано функции хэширования. Однако в некоторых случаях, когда структура проекта относительно сложна и содержит иерархию введенных классов и функцийэ, пользоваться семантическим деревом нецелесообразно. Для таких случаев разработан анализатор файла, который отделяет необходимую информацию с целью осуществления дальнейших действий (рис.5).

С целью классификации подгрупп выбранных тестов по критической степени предложен программный декоратор-шаблон, структура которого приведена на рис.6. Структура представленного шаблона позволяет ввести дополнение к уже имеющейся функциональности, прежде чем начнется запуск основной функциональности. Этот подход применялся для классификации тест-кейсов. Классификация проводится по критическим степеням. В процессе прогона/запуска тестов проводится декорация данного тест-кейса, в ходе которой и проверяется критическая степень. Если критическая степеньй удовлетворяет заранее выбранному промежутку, то декоратор продолжает работу по основной функциональности тест-кейса, в противном случае - пропускается осуществление

основной функциональности тест-кейса. Подобным образом запускаются только те тест-кейсы, которые удовлетворяют критической степени.

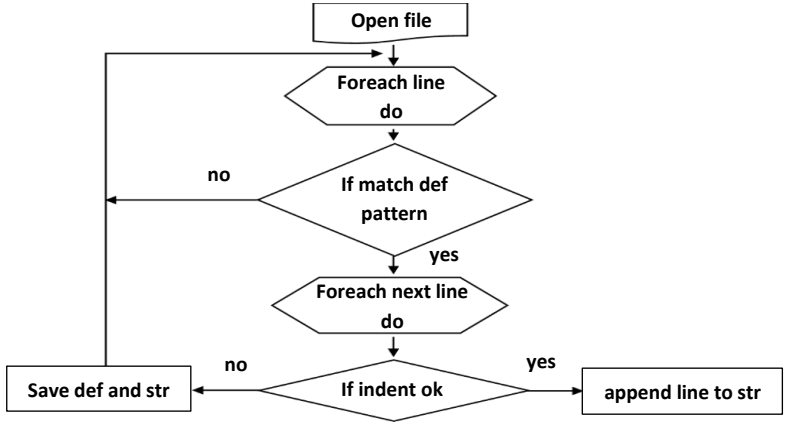

Рис.5. Сбор информации, необходимой для функций

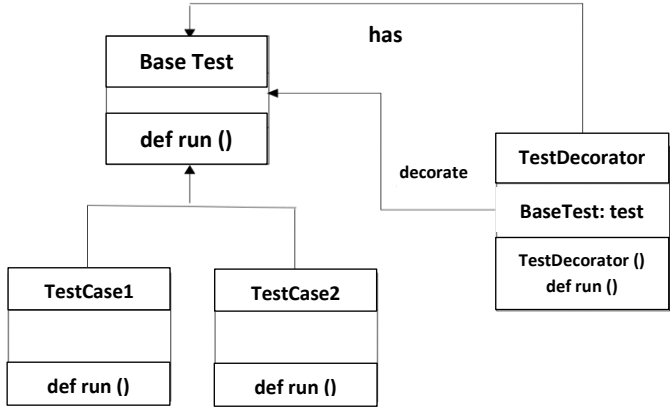

Рис.6. Структурный декоратор-шаблон

Четвертая глава посвящена разработке ПО средств регрессионного тестирования САПР электронных схем. С целью осуществления описанного в предыдущей главе и процесса отбора регрессионных тестов, разработано программное средство, которое позволяет проводить отбор регрессионных тестов

исходя из принципов гибридного механизма. На рис.7 отображен интерфейс программного инструмента.

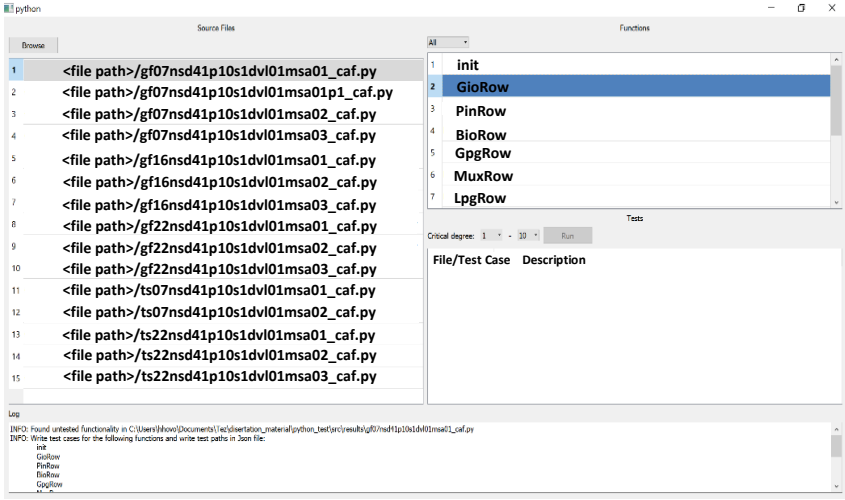

Рис.7. Интерфейс программного средства

Общая работа программного средства представлена на рис.8, где приведена блок-схема процесса проведения отбора регрессионных тестов.

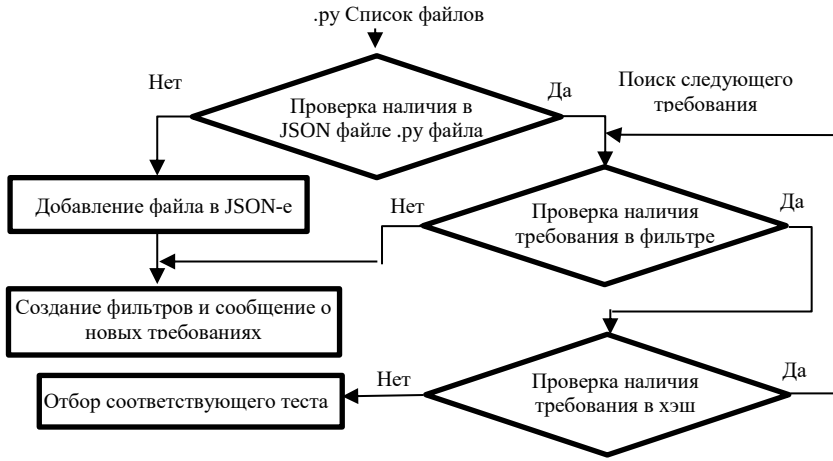

Рис.8. Блок-схема процесса отбора регрессионных тестов

В ходе исследований были использованы и подвергнуты изменениям CAF описания представления схемы реального проекта. В табл. 2 представлена зависимость срока выявления тестов от количества имеющихся в проекте функций. В таблице представлены данные работы программы. В левой части приведены наименования файлов, используемых в проекте, каждый из которых содержит от 13 до 20 функций. Тело каждой функции состоит из 100-200 строк. Второй столбец представляет время обнаружения имеющихся в файле функций, а третий столбец-время обнаружения тестов. Последний столбец описывает полученное суммарное время.

Таблица 2

Время (сек.) обнаружения и отбора регрессионных тестов в зависимости от количества функций

Наименование файла	Обнаружение функций	Отбор тестов	Общее время
gf07nsd41p10s1dvl01msa01_caf.py	0.19942	0.001021	0.200441
gf07nsd41p10s1dvl01msa01p1_caf.py	0.26926	0.001994	0.271254
gf07nsd41p10s1dvl01msa02_caf.py	0.19831	0.002142	0.200452
ts07nsd41p10s1dvl01msa01_caf.py	0.21743	0.001834	0.219264
ts07nsd41p10s1dvl01msa02_caf.py	0.19276	0.001548	0.194308
ts22nsd41p10s1dvl01msa01_caf.py	0.18997	0.001756	0.191726
ts22nsd41p10s1dvl01msa02_caf.py	0.17724	0.001833	0.179073
ts22nsd41p10s1dvl01msa03_caf.py	0.22013	0.002011	0.222141
gf16nsd41p10s1dvl01msa02_caf.py	0.19576	0.001793	0.197553
gf16nsd41p10s1dvl01msa03_caf.py	0.19882	0.001955	0.200775

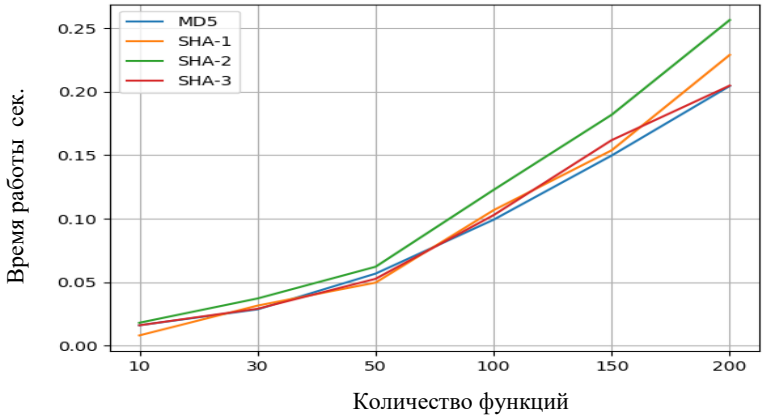

Рис.9. Зависимость времени работы хэш-функций от количества функций

Из показателей, приведенных на рис.9, видно, что при работе 4-ех алгоритмов получены сравнительные результаты. Однако после некоторых значений количества функций алгоритм MD5 превосходит по скорости остальные хэширующие функции.

ОСНОВНЫЕ ВЫВОДЫ ПО ДИССЕРТАЦИОННОЙ РАБОТЕ

1. В результате анализа текущих реалий тестирования САПР электронных схем и вызовов развития обоснована актуальность проблем внедрения новых средств регрессионного тестирования в системах автоматизации электронного проектирования и их непрерывного совершенствования, вызванных усложнением проектов и развитием функциональных возможностей [1].

2. Разработан основанный на выявлении изменений механизм отбора регрессионных тестов с применением алгоритма хэширования для сокращения времени тестирования [2,3].

3. Разработан механизм отбора регрессионных тестов с внедрением фильтра Блума, что привело к повышению эффективности регрессионного тестирования в САПР электронных схем и ускорению поиска данных [5].

4. Разработана новая структура хранения данных, которая учитывает недостатки использования матрицы прослеживаемости в случае крупномасштабной обработки данных. Разработанная структура основана на использовании формата JSON, применяемого для обновления и удаления хранилища данных [4, 5].

5. Реализован алгоритм, основанный на линейном сравнении выявления изменений, внесенных в новый проект, с использованием алгоритмов линейного сравнения Хэмминга и Ратклиффа/Обершельпа. Использование линейного сравнения позволяет найти точки, где были внесены изменения [статья в печати].

6. Результаты исследования показали, что в случае регрессионного тестирования в САПР электронных схем механизм отбора регрессионных тестов на основе алгоритма хэширования, в зависимости от сложности функции, работает в 11÷37 раз быстрее, чем механизм отбора регрессионных тестов, основанный на алгоритме линейного сравнения Хэмминга, проявляя определенную вероятность столкновения хеш-значений. Согласно полученным данным, вероятность столкновения хэш-значений достигает 0,5-а в случае довольно большого количества хэшей - 77163 [1, 2].

Основные результаты диссертации опубликованы в следующих 5-и работах, и еще одна статья в настоящее время находится в печати. Две из опубликованных работ включены в базу данных Scopus, 2 без соавторов. Опубликованные работы:

1. Hakobyan Hovhannes H., Vardumyan Arman V., Kostanyan Harutyun T. Unit Regression Test Selection According to Different Hashing Algorithms. 2019 IEEE East-West Design & Test Symposium (EWDTS). - 2019 - P. 59-62.
2. Melikyan Vazgen Sh., Hakobyan Hovhannes H., Kaplanyan Taron K., Momjyan Arsen M. Unit Regression Test Selection Mechanism Based on Hashing Algorithm. 2019 IEEE East-West Design & Test Symposium (EWDTS). – 2019 – P. 103-107.
3. Hakobyan H.H., Testing Memory Compilers Using the Regression Test Selection Mechanism // Proceedings of NPUA: Information Technologies, Electronics, Radio engineering. 2019. № 2. – P. 45-53.
4. Հակոբյան Հ. Հ., Մոդուլային ռեգրեսիոն թեստերի ընտրությունը հետագծելիության մատրիցի կիրառմամբ – Հայաստանի ճարտարագիտական ակադեմիայի Լրաբեր (ՀՃԱԼ). - 2019. - Հ. 16, N2 – էջ 243-247:
5. Հակոբյան Հ.Հ., Հարությունյան Ա.Գ., Ծրագրային համակարգերի ռեգրեսիոն թեստավորման արդյունավետության բարձրացումը Բլումի ֆիլտրի կիրառմամբ .- ՀԳԱԱ և ՀԱՊՀ տեղեկագիր, տեխնիկական գիտությունների սերիա.- 2020. - Հ. 73, N1 – էջ 101-107:

В настоящее время находится в печати следующая статья:

Hakobyan H.H., Kostanyan H.T., Vardumyan A.V., Harutyunyan A.G. Unit Regression Test Selection Based on Several String-Matching Algorithms // Proceedings of NPUA: Information Technologies, Electronics, Radio engineering. 2020. № 1 (в печати).

ՀԱԿՈԲՅԱՆ ՀՈՎՀԱՆՆԵՍ ՀԱՄԼԵՏԻ

ԷԼԵԿՏՐՈՆԱՅԻՆ ՍԽԵՄԱՆԵՐԻ ԱՎՏՈՄԱՏԱՑՎԱԾ ՆԱԽԱԳԾՄԱՆ ՀԱՄԱԿԱՐԳԵՐԻ ՌԵԳՐԵՍԻՈՆ ԹԵՍՏԱՎՈՐՄԱՆ ՄԻՋՈՑՆԵՐԻ ՄՇԱԿՈՒՄԸ

ԱՄՓՈՓԱԳԻՐ

Ժամանակակից միկրոէլեկտրոնային սարքերի դինամիկ զարգացման պայմաններում անհրաժեշտություն է առաջանում՝ անընդհատ կատարելագործելու դրանց թեստավորման միջոցները: Դրա հետևանքով, ԻՍ-երի ֆունկցիոնալության անընդհատ զարգացմանը զուգընթաց, դրանց թեստավորման միջոցների մեջ աճում են ռեգրեսիոն թեստավորման ծավալները: Ռեգրեսիոն թեստավորումն իրականացվում է նախագծում կատարված փոփոխություններից հետո՝ նպատակ ունենալով ստուգելու նոր տարբերակի աշխատունակությունը: Ռեգրեսիոն թեստավորման բնագավառում ներկայումս հայտնի է նախագծերի թեստավորման չորս եղանակ՝ ամբողջական թեստավորում, թեստերի ընտրություն, թեստերի դասակարգում, հիբրիդ:

Չնայած նշված խնդրի լուծման ուղղությամբ ձեռք բերված որոշակի հաջողություններին, նախագծերի բարդության աճով և ֆունկցիոնալության զարգացմամբ պայմանավորված, հրատապ խնդիրներ են դառնում էլեկտրոնային նախագծման ավտոմատացման համակարգերում ռեգրեսիոն թեստավորման նոր միջոցների ներդրումը և դրանց անընդհատ կատարելագործումը: Մասնավորապես՝ հրատապ է դառնում նախագծման բարդացման հետ մեկտեղ՝ նախագծվող օբյեկտի աշխատանքի կայունության ապահովման խնդիրը: Այլ կերպ ասած, պահանջվող ֆունկցիոնալ նոր ծավալների ավելացումը և սխալների ուղղումը չպետք է բացասաբար ազդեն ընթացիկ նախագծի աշխատունակության վրա:

Թեստավորման առկա միջոցների թերությունների շտկման նպատակով առաջարկվում են ռեգրեսիոն թեստերի ընտրության մեխանիզմներ՝ հիմնված հեշավորման ալգորիթմների վրա: Առաջարկվում է նաև, թեստերի ընտրության նպատակով, հետագծելիության մատրիցի փոխարեն օգտագործել մշակված նոր կառուցվածքը, որը հիմնված է տվյալների պահպանման JSON ձևաչափի վրա: Առաջարկվում է թեստերի ընտրության հիբրիդ մեխանիզմում թեստերի դասակարգման գործընթացը կատարել կրիտիկական աստիճանների հաշվառմամբ՝ օգտագործելով դեկորատոր կառուցվածքային ձևանմուշը:

Հետազոտության առարկան է ԻՍ-երի ներկայացման տեքստային նկարագրման ֆունկցիոնալության բարդացման, նախագծի ընդլայնման և փոփոխման հաշվառմամբ՝ ԻՍ-երի թեստավորման, ռեգրեսիոն թեստերի ընտրության մեթոդական, ալգորիթմական և ծրագրային ապահովման միջոցները:

Աշխատանքի նպատակն է ԻՍ-երում նախագծի ֆունկցիոնալության փոփոխման պայմաններում ռեգրեսիոն թեստավորման արդյունավետ մեթոդների, ալգորիթմական և ծրագրային գործիքների մշակումը և հետազոտումը:Ծրագրային միջոցի շնորհիվ հնարավոր է եղել ընտրված թեստ-փաթեթից առանձնացնել այն թեստ-դեպքերը, որոնք առավել կարևոր են, այսինքն ունեն՝ բարձր կրիտիկական աստիճան: Այս կրկնակի զտման մեխանիզմն իրականացվել է կառուցվածքային դեկորատոր ձևանմուշի կիրառմամբ: Ատենախոսության կատարման ընթացքում օգտագործվել են ԻՍ-երի ներկայացման տեքստային նկարագրման ֆայլերը, որոնց միջոցով նկարագրվում են հիշասարքերի ֆունկցիոնալությունները:

- Առաջարկվել է նախագծում կատարված ֆունկցիոնալ փոփոխության հայտնաբերման եղանակ՝ հիմնված հեշավորման ալգորիթմների վրա: Էլեկտրոնային սխեմաների ԱՆՀ-երում նախագծային փոփոխությունների

հայտնաբերման աշխատանքի ժամանակը կրճատելու նպատակով՝ կատարված հետազոտությունների արդյունքում ընտրվել է առավել արագագործ հեշավորման ալգորիթմը:

- Առաջարկվել է տվյալների պահպանման նոր կառուցվածք՝ հիմնված JSON ձևաչափի վրա, որը թեստ-փաթեթների ծավալների մեծացման պայմաններում ապահովում է էլեկտրոնային սխեմաների ԱՆՀ-երում դրանց իրագործման արդյունավետությունը:

- Առաջարկվել է, մեծածավալ տվյալներում թեստերի որոնման աշխատանքի ժամանակը կրճատելու նպատակով, ռեգրեսիոն թեստերի ընտրման մեխանիզմում Բլումի ֆիլտրի ներդրման վրա հիմնված որոնման համակարգ:

- Առաջարկվել է կրիտիկական աստիճանների վրա հիմնված դասակարգման մեխանիզմ, որն ապահովում է ընտրված թեստ- փաթեթներում թեստ-դեպքերի կրկնակի զտում: Մշակված մեխանիզմի հիմքում ընկած է ծրագրային կառուցվածքային դեկորատոր ձևանմուշը:

HAKOBYAN HOVHANNES HAMLET

REGRESSION TESTING OF AUTOMATED SYSTEMS OF ELECTRONIC CIRCUITS

SUMMARY

With the dynamic development of modern microelectronic devices, there is a need of a constant improvement in their testing methods. As a result of the continuous enlargement of IC functionalities, the proportion of the regression tests in the IC testing methods group is expanding. Regression testing is performed once the changes are implemented in the software, in order to test the performance of the new version. The following four methods mostly used in regression testing regression testing: complete testing, selective testing, classified testing and hybrid. It can be stated that despite certain achievements in solving the mentioned problems, due to the increase of the complexity of the projects and their functional enhancements, the introduction of new regression testing methods in electronic design automated systems and their constant improvement is becoming a high priority issue. In order to correct the drawbacks of the existing testing tools, mechanisms for selecting regression tests based on hashing algorithms are introduced. It is also recommended to use a new structure based on the JSON data storage format instead of the traceability matrix to select the tests. It is suggested

to perform the test classification process in the hybrid mechanism of test selection by calculating critical degrees using a decorative structural sample. The subject of the research is the methodological, algorithmic and software means of IC testing, selection of regression tests, taking into account the complexity of the functionality of text description, project expansion and modification of IC presentation. The objective of the work is to develop and study effective methods of regression testing, algorithmic and software tools in the conditions of project functionality changes in IC. Thanks to the software, it was possible to separate from the selected test package the tests that are the most important, that is, have a high critical level. This double filtration mechanism was implemented using a structural decorative pattern. In the course of the dissertation, the text description files for the presentation of ICs were used, which describe the functionality of the memory devices.

• A method for detecting functional changes in the design based on hashing algorithms has been proposed. In order to reduce the time needed to detect changes in the automated design systems of electronic circuits, the fastest hashing algorithm was selected as a result of the research.

• A new data storage structure has been developed based on JSON format, which ensures the efficiency in automated design systems of electronic circuits, of their operation in the conditions of increasing the number of test packages.

• In order to reduce the time required to search for tests in larger data, a search engine has been proposed based on the introduction of Bloom filter in the mechanism for selecting regression tests.

• A critical grading mechanism has been proposed that provides double refinement of test cases in the selected test packages. The developed mechanism is based on the programmatic structural decorative pattern.