DOI: 10.15514/ISPRAS-2026-38(3)-13

Оценка эффективности по критерию трудоемкости применения тестовых систем на основе генераторов конвейеризированных контрольных кодов

М.О. Ратников, ORCID: 0009-0006-2540-1873<m.o.ratnikov@mail.ru>

Московский авиационный институт, МАИ,
Россия, 125993, г. Москва, Волоколамское шоссе, д. 4.

Аннотация. Данная статья посвящена оценке трудоемкости разработки тестовых систем на основе конвейеризированных генераторов контрольных кодов и эффективности использования подобных тестовых систем для выбора способа обеспечения сбоеустойчивости системы на базе ПЛИС с точки зрения трудоемкости всего проекта. Для оценки трудоемкости предлагается использовать модель COCOMO II, коэффициенты которой адаптированы для HDL-кода. В статье рассмотрен пример имплементации трех способов обеспечения сбоеустойчивости в разработанную систему на базе ПЛИС. Была сделана оценка трудоемкости разработки защищенной системы для двух маршрутов проектирования. В первом случае было разработано три версии целевой защищенной системы, после чего была выбрана лучшая по заданным критериям. Во втором случае до начала разработки защищенной целевой системы был проведен дополнительный анализ с использованием тестовой системы на основе конвейеризированного генератора контрольных кодов, в ходе которого был выбран лучший по заданным характеристикам способ обеспечения сбоеустойчивости, который и был имплементирован в целевую систему. Далее было проведено сравнение трудоемкостей этих маршрутов проектирования и сделан вывод об эффективности применения методики выбора способа обеспечения сбоеустойчивости на основе использования тестовой системы. Оценка эффективности по критерию трудоемкости была проведена на основе количества строк кода в проекте и состояла из нескольких последовательных шагов: от оценки количества строк кода в проекте целевого устройства без имплементации методов обеспечения сбоеустойчивости и целевого устройства, защищенного каждым из рассмотренных методов обеспечения сбоеустойчивости, до сравнения изменения трудоемкости этапа выбора и имплементации способа обеспечения сбоеустойчивости и общего изменения трудоемкости проекта.

Ключевые слова: программируемые логические интегральные схемы (ПЛИС); модель COCOMO; трудоемкость; оценка затрат; контрольный код; генератор контрольных кодов MD4; обеспечение сбоеустойчивости; тестовая система; тестирование ПЛИС.

Для цитирования: Ратников М.О. Оценка эффективности по критерию трудоемкости применения тестовых систем на основе генераторов конвейеризированных контрольных кодов. Труды ИСП РАН, том 38, вып. 3, часть 1, 2026 г., стр. 209–222. DOI: 10.15514/ISPRAS–2026–38(3)–13.

Performance evaluation based on the cost criterion of test systems using pipelined control code generators

M.O.Ratnikov, ORCID: 0009-0006-2540-1873<m.o.ratnikov@mail.ru>

Moscow, Moscow Aviation Institute, MAI,
Russia, 125993, Moscow, Volocolamskoe highway, 4.

Abstract. This paper is devoted to evaluating the development effort required for designing test systems based on pipelined control code generators and to assessing the effectiveness of using such test systems for selecting fault-tolerance mechanisms in FPGA-based systems from the standpoint of overall project effort. To estimate development effort, the COCOMO II model is proposed, with its coefficients adapted for HDL code. The paper presents an example implementation of three different fault-tolerance approaches within a developed FPGA-based system. The development effort for creating a fault-tolerant system was evaluated for two design routes. In the first route, three versions of the target fault-tolerant system were implemented, and the best one was selected according to predefined criteria. In the second route, before developing the protected target system, an additional analysis was performed using a test system based on a pipelined control code generator. During this analysis, the optimal fault-tolerance method was identified according to specified characteristics and then implemented in the target system. A comparison of the design effort between these two routes was conducted, leading to a conclusion about the efficiency of using the proposed test-system-based approach for selecting fault-tolerance mechanisms. The efficiency evaluation, in terms of development effort, was based on the number of lines of code in the project and included several consecutive steps: from estimating the code size of the baseline target device (without fault-tolerance mechanisms) and the same device protected by each of the fault-tolerance methods under consideration, to comparing the changes in the effort required for selecting and implementing the fault-tolerance approach, as well as the overall change in total project effort.

Keywords: FPGA; COCOMO; effort estimate; project cost; control code; MD4; fault-tolerance; test system; FPGA testing.

For citation: Ratnikov M.O. Performance evaluation based on the cost criterion of test systems using pipelined control code generators. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 3, part 1, 2026, pp. 209-222 (in Russian). DOI: 10.15514/ISPRAS-2026-38(3)-13.

1. Введение

К надежности и сбоеустойчивости современных вычислительных систем, работающих в условиях негативных внешних воздействий, например, системам управления космических аппаратов и медицинской техники, авионике, промышленным системам управления, предъявляют особые требования, так как отказ или даже, в отдельных случаях, кратковременный сбой может привести к критическим последствиям. При этом, к этим устройствам могут предъявляться дополнительные требования по скорости работы, энергопотреблению, тепловыделению и так далее. Задача разработки подобных систем усложняется тем, что такие устройства часто являются малосерийными. Сочетание перечисленных факторов очень сильно усложняет задачу выбора аппаратной платформы. Решения на основе универсальных микропроцессоров не всегда могут одновременно выполнить требования по производительности и энергопотреблению, разработка специализированных СБИС и БМК дорого стоит, занимает много времени и не допускает исправлений в готовых изделиях [1]. Поэтому одной из самых широко распространенных платформ для разработки встраиваемых систем являются ПЛИС.

Проектирование отказоустойчивых систем на ПЛИС является нетривиальной задачей, при решении которой необходимо найти компромисс между уровнем надежности, потреблением энергии, занимаемой площадью кристалла и временем разработки. Наивная реализация тех или иных подходов к обеспечению сбоеустойчивости может приводить к чрезмерному увеличению используемых ресурсов и недостаточной эффективности обработки различных типов отказов. Задача выбора подходящего способа обеспечения сбоеустойчивости часто

усложняется тем, что этот выбор нужно сделать на раннем этапе разработки устройства, до того, как будет завершена разработка функциональной нагрузки ПЛИС. В противном случае приходится имплементировать в разработанную полнофункциональную (целевую) систему различные способы обеспечения сбоеустойчивости и сравнивать их аналитически или в процессе испытаний, что увеличивает трудоемкость проекта и стоимость разработки.

Большинство современных методик выбора способа обеспечения сбоеустойчивости, используемых на ранних этапах разработки, основывается на разработке специализированной тестовой системы и использовании специального программного обеспечения для проведения анализа. Кроме того, во время разработки устройства на базе ПЛИС возникает ряд задач, связанных с тестированием самой ПЛИС и ее функционального окружения. Современные методики тестирования обычно предполагают создание ряда тестовых функциональных описаний для конфигурирования ПЛИС, каждое из которых задействуется на своем этапе. Это усложняет и увеличивает длительность процесса разработки устройства. Также, необходимо отметить, что применяемые в настоящее время подходы к разработке тестовых систем имеют ряд недостатков, связанных, например, с масштабируемостью системы и обнаружением сбоев. Одним из способов решения данной проблемы является использование универсальной тестовой системы на основе конвейеризированного генератора контрольного кода [2]. Подобные тестовые системы могут быть использованы на разных этапах разработки, и позволяют быстро сконфигурировать и адаптировать тестовую систему для решения различных задач тестирования на базе ПЛИС.

Представляемое в данной статье исследование посвящено оценке трудоемкости разработки тестовых систем на основе конвейеризированных генераторов контрольных кодов и эффективности использования подобных тестовых систем для выбора способа обеспечения сбоеустойчивости системы на базе ПЛИС с точки зрения трудоемкости всего проекта.

2. Тестовые системы ПЛИС

К тестовым системам, которые могут быть использованы для проведения сравнения различных методов обеспечения сбоеустойчивости и проведения тестирования системы на базе ПЛИС, предъявляется ряд требований, связанных с: обеспечением корректного проведения этапов синтеза, трассировки и генерации конфигурационного файла для целевой ПЛИС; обеспечением максимальной чувствительности к сбоям и отказам, в том числе многократным; обеспечением быстрого изменения степени логической загрузки ПЛИС без значительных исправлений исходного кода; обеспечением соответствия между предполагаемым значением занятых логических ресурсов и значением, полученным с помощью оценки синтезатора и трассировщика; обеспечением возможности вычисления значений, получаемых в процессе работы тестового устройства, заранее, и, соответственно, сравнения с данными, полученными в результате работы исследуемой ПЛИС.

Наиболее полно этим требованиям соответствуют конвейерные системы, в которых каждая стадия состоит из набора логических элементов, реализующих выбранную синтезируемую функцию, и регистра для хранения вычисленного значения. Такие системы хорошо масштабируются путем увеличения количества стадий конвейера, позволяют не допускать усложнения логических функций, реализованных на каждой из ступеней. Также преимуществом конвейерных систем с повторяющимися функциями является их регулярная структура, что значительно упрощает работу по размещению логической нагрузки внутри ПЛИС [3].

Конвейеризация алгоритма вычисления контрольного кода – это представление циклического алгоритма в виде конвейерной структуры, в которой на каждой ступени реализуется очередная итерация (или часть итерации) цикла алгоритма генерации контрольного кода. Таким образом ступень с номером i получает промежуточные данные и необходимые признаки от i–1 ступени, после чего в соответствии с выбранным алгоритмом

выполняет вычисления на основе полученных данных и значения i-го бита входного потока. Вычисленные промежуточные значения и признаки передаются на i+1-ю стадию. Данные с последней стадии передаются на выходы тестовой системы. После окончания разгона такого конвейера на каждом такте работы тестовой системы на выход передается очередное значение контрольного кода.

В качестве логической нагрузки могут быть выбраны любые алгоритмы, позволяющие выполнять побитную обработку входных данных, например, хэш-функции или алгоритмы генерации самокорректирующихся кодов. Значения обрабатываемых бит могут быть заданы: константами, значениями, установленными на входах ПЛИС (статическими или динамическими), результатом работы внутренних блоков ПЛИС (в том числе блоков самотестирования). Данные от этих устройств передаются на тестовую систему и образуют входной поток данных. Перед началом тестирования вычисляется эталонный выходной поток – набор контрольных кодов, вычисленных тестовой системой при ожидаемом «правильном» входном потоке. Из-за сбоев, некорректной работы внешних устройств и самой ПЛИС реальный выходной поток может отличаться от эталонного.

Сравнение полученных результатов выполняет внешний по отношению к тестовой системе блок – анализатор, результатом работы которого является признак наличия ошибки или синдром ошибки.

Характерной особенностью конвейерных систем является обновление значений в запоминающих элементах на каждом такте работы устройства. Таким образом, при появлении сбоя в тестовой системе, через несколько тактов изменится выходное значение, а после исправления сбоя, значение на выходах вернется к ожидаемому.

Если среди требований, предъявляемых к тестовой системе, есть требование не только обнаружения сбоя, но и определения места его возникновения, то используется тестовая система на основе конвейеризированного алгоритма вычисления самокорректирующихся кодов. Такая тестовая система может быть применена не только для тестирования устройств на базе ПЛИС, но и для выбора метода обеспечения сбоеустойчивости целевой системы. Методика выбора способа обеспечения сбоеустойчивости с помощью тестовой системы на базе конвейеризированного генератора контрольного кода основана на следующих особенностях таких тестовых систем:

- регулярная структура из повторяющихся блоков, которая позволяет минимизировать трудоемкость имплементации анализируемого метода в тестовую систему;
- масштабируемость: размер тестовой системы легко изменяется, что позволяет максимально точно приблизить затраты ресурсов ПЛИС на реализацию незащищенной тестовой системы к ожидаемым затратам на незащищенную целевую систему;
- возможность выбора алгоритма контрольного кода, лежащего в основе тестовой системы, что также дает возможность приблизить характеристики незащищенной тестовой системы к ожидаемым характеристикам незащищенной целевой системы.

Таким образом, появляется возможность еще до окончания разработки целевой системы подобрать похожую по ожидаемым характеристикам тестовую систему, имплементировать в нее различные методы обеспечения сбоеустойчивости и оценить возможные характеристики целевой системы, защищенной с помощью того или иного метода обеспечения сбоеустойчивости. После выбора наиболее подходящего в конкретных условиях метода, можно выполнить реализацию целевой системы [4]. Эта же тестовая система без дополнительных доработок применима как для входного тестирования ПЛИС, так и для проверки корректности работы компонентов, которые входят в разрабатываемую систему и взаимодействуют с ПЛИС (линии соединения, подсистемы питания и охлаждения и так

далее), то есть описанная выше тестовая система является более универсальной по сравнению с наиболее распространенными в настоящее время тестовыми системами.

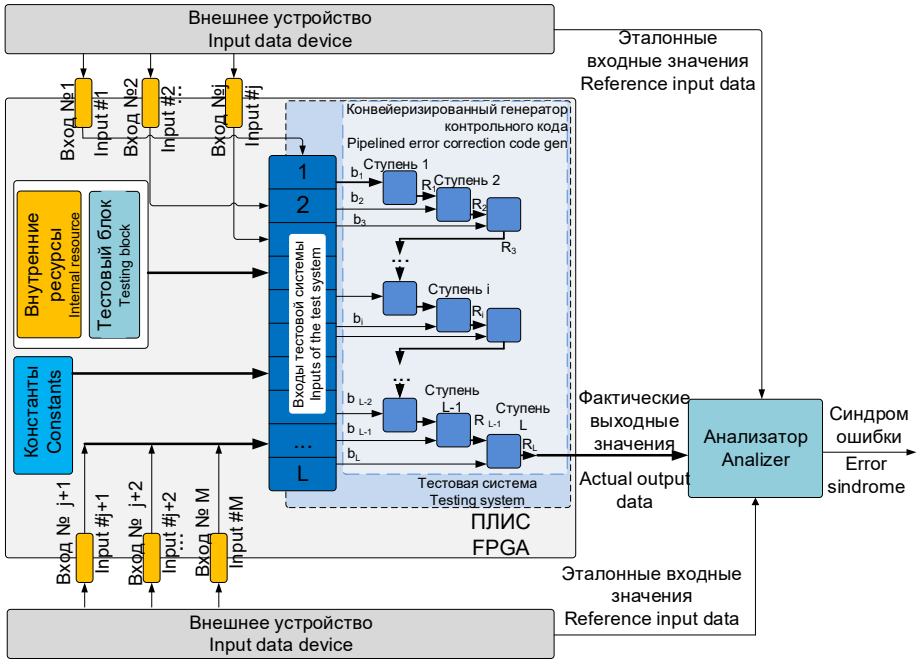

Рис. 1. Общий вид тестовой системы
Fig. 1. General view of the testing system

3. Оценка трудоемкости проекта

Трудоёмкость – количество рабочего времени человека, затрачиваемое на производство единицы продукции. Трудоёмкость обратно пропорциональна показателю производительности труда (количеству продукции, вырабатываемой за единицу рабочего времени) [5].

3.1 Модель трудоемкости COCOMO II

Среди алгоритмических моделей одной из наиболее проработанных и широко известных является модель COCOMO II [6] (Constructive Cost Model II), представляющая собой эволюционное развитие классической модели Барри Боэма. COCOMO II – это параметрическая модель, которая оценивает трудозатраты (в человеко-месяцах) и сроки разработки на основе оценки размера проекта (в тыс. строк кода или функциональных точках) и учета множества cost drivers – факторов стоимости, которые отражают характеристики продукта, платформы, персонала и проекта в целом. С момента своей первой публикации в 2000 году она много раз дорабатывалась и уточнялась и до сих пор остается актуальной и находит широкое применение [7-9].

Полная формула модели COCOMO II имеет следующий вид [7]:

$$\text{Затраты} = \text{A} \times \text{K}_{\text{req}} \times \text{Размер}^{\text{B}} \times \text{M}_{\text{e}} + \text{Затраты}_{\text{auto}}\ [\text{чел.}-\text{мес.}],$$

где A – коэффициент для масштаба, определенный разработчиком модели (2,94), K_{req} – коэффициент, отражающий вероятность изменения требований к продукту, Размер – количество строк кода, B – общий фактор издержек, M_e – множитель поправки, учитывающий факторы формирования затрат, $Затраты_{auto}$ – затраты на код, который генерируется автоматически. При этом:

$$B = 0{,}91 + 0{,}01 \times \sum_{i=1}^{5} W_i,$$
$$\text{M}_e = \prod_{i=1}^{17} EM_i,$$

где коэффициенты W_i задают факторы масштаба (прецендентность, гибкость процесса разработки, архитектура, сработанность команды, зрелость процессов), а EM_i – факторы формирования затрат. Для каждого из этих параметров определены от 4 до 6 уровней от «очень низкий» до «критический», которые выбираются в зависимости от конкретного проекта и команды разработчиков. Числовые значения выбираются из таблиц приведенных в [6]. Вычислить итоговую трудоемкость проекта с помощью модели COCOMO II можно как самостоятельно, так и с использованием специализированных калькуляторов [8].

Характерной особенностью модели COCOMO II является возможность сравнения трудозатрат на схожие по сложности проекты, выполняемые одними и те же разработчиками, простым вычислением отношения трудозатрат на оба проекта. Например, для двух проектов, отличающихся только объемом разрабатываемого кода без использования автогенерируемого кода, можно вычислить следующее отношение:

$$\text{относительные затраты} = \frac{\text{Затраты 1}}{\text{Затраты 2}} = \frac{\text{A} \times \text{K}_{\text{req}} \times \text{Размер}^{\text{B1}} \times \text{M}_{\text{e}}}{\text{A} \times \text{K}_{\text{req}} \times \text{Размер}^{\text{B2}} \times \text{M}_{\text{e}}},$$

следовательно:

$$\text{относительные затраты} = \frac{\text{Размер}^{\text{B1}}}{\text{Размер}^{\text{B2}}}.$$

3.2 Особенности применения COCOMO II для оценки трудоемкости HDL-проектов

Несмотря на то, что изначально модель COCOMO II разрабатывалась для программных проектов, она, наряду с другими моделями и метриками, основанными на оценке LOC (Lines Of Code – количество эквивалентных строк исходного кода в проекте), может успешно применяться и для оценки трудоемкости при разработке аппаратуры [7-9]. Общие рекомендации по работе с моделью COCOMO II для этих целей сводятся к уточнению основных множителей и коэффициентов данной модели с учетом специфики разработки аппаратуры. В частности, при оценке ожидаемого объема эквивалентного кода рекомендуется разделить код по назначению: автогенерируемый код, RTL-код и код системы верификации с учетом различного количества трудозатрат на строку кода.

Также необходимо отметить, что при оценке трудоемкости проектов, связанных с разработкой HDL при помощи модели COCOMO II, оцениваются трудозатраты на разработку только исходного кода проекта, а значит, при планировании необходимо отдельно учитывать трудоемкости этапов разработки топологии, отладки готового устройства и так далее.

В табл. 1 приведены основные коэффициенты, необходимые для вычисления трудоемкости по модели COCOMO II.

Табл. 1. Коэффициенты модели COCOMO II для HDL.
Table 1. COCOMO II drivers for HDL projects.

Характеристика	Описание	HDL-адаптация и комментарии	Возможные значения
Precedentedness	Прецедентность, опыт в подобных проектах, есть ли аналоги	HDL-проекты редко полностью типовые	низкие – номинальные
Development Flexibility	Гибкость процесса разработки и жесткость требований	Есть ряд ограничений (тайминг системы, объем ресурсов), которые нельзя отменить или изменить	номинальные
Architecture / Risk Resolution	Архитектура и разрешение рисков	Архитектура проекта обычно определяется на раннем этапе, как и основные интерфейсы	номинальные – высокие
Team Cohesion	Согласованность команды	Зависит от наличия всех необходимых специалистов в команде	номинальные
Process Maturity	Зрелость процессов	Зависит от наличия управления версиями, регрессионного тестирования, CICD, ревью исходных кодов	номинальные – высокие
Required Software Reliability	Требуемая надежность ПО	Зависит от решаемых задач и цены ошибки при разработке. Для систем, где требуется обеспечение сбоеустойчивости – высокая	высокие
Data Base Size	Размер базы данных	Объем данных, обрабатываемый системой. Обычно средний	номинальные
Product Complexity	Алгоритмическая и структурная сложность продукта	Общий параллелизм, конечные автоматы, борьба с метастабильностью	высокие – очень высокие
Developed for Reusability	Разработка с учетом повторного использования	Номинальная для обычных проектов, высокая для отчуждаемых IP	номинальные
Documentation Match to Lifecycle Needs	Соответствие документации потребностям жизненного цикла	Аналогично документации для программных продуктов	номинальные
Analyst Capability	Квалификация аналитиков	Квалификация архитекторов. Обычно номинальная или высокая	номинальные
Programmer Capability	Квалификация программистов	Квалификация разработчиков	номинальные
Personnel Continuity	Постоянство персонала	Аналогично модели для программных продуктов	номинальные
Application Experience	Опыт работы в предметной области	Зависит от проекта и конкретного коллектива разработчиков	номинальные
Platform Experience	Опыт работы с платформой	Знание конкретной платформы и маршрута разработки для нее	номинальные
Language and Toolset Experience	Опыт работы с языками и инструментами разработки	Знание основных инструментов: симулятора, синтезатора	номинальные
Time Constraint	Ограничение по времени выполнения	Требования к временным характеристикам целевой системы	высокие – очень высокие
Storage Constraint	Ограничение по объему памяти	В том числе ограничение по ресурсам микросхемы	номинальные – высокие
Platform Volatility	Изменчивость платформы	Смена инструментария или аппаратной платформы. Зависит от проекта	номинальные
Use of Software Tools	Использование инструментов разработки	Инструменты для разработки систем на базе ПЛИС зависят от платформы	номинальные
Multisite Development	Распределенная разработка	Аналогично модели для программных продуктов	номинальные
Required Development Schedule	Требуемое расписание разработки	Сжатость графика. Зависит от проекта, например, при разработке ASIC требования более жесткие из-за необходимости заранее определиться с датой начала производства микросхем на фабрике	номинальные – очень высокие

4. Оценка эффективности по критерию трудоемкости

Предлагается провести сравнение реализации одного и того же проекта сбоеустойчивого целевого устройства: во-первых, с применением методики выбора способа обеспечения сбоеустойчивости с использованием тестовой системы на основе конвейеризированного генератора контрольных кодов, и во-вторых, без применения данной методики.

Предполагается, что при применении предложенной методики берется заранее разработанная незащищенная тестовая система, затем на ее основе разрабатываются защищенные тестовые системы – по одной на каждый из исследуемых методов сбоеустойчивости. После проведения анализа, в ходе которого определяется лучший по критериям вероятности появления сбоя и реализуемости способ, разрабатывается и верифицируется сначала незащищенная целевая тестовая система. Далее на ее основе разрабатывается одна версия защищенной тестовой системы с использованием выбранного ранее способа обеспечения сбоеустойчивости.

В случае, когда методика не применяется, предполагается, что разрабатывается и верифицируется незащищенная целевая система, на основе которой разрабатывается несколько защищенных целевых систем, в каждой из которых реализован один из анализируемых методов обеспечения сбоеустойчивости. Далее проводится анализ с целью определения защищенной тестовой системы с лучшими характеристиками по критериям вероятности появления сбоев и реализуемости.

4.1 Методика сравнения

Предлагается следующая методика сравнения, заключающаяся в проведении оценки на основе количества строк кода в проекте. Методика оценки состоит из следующих шагов:

1) оценить количество строк кода в проекте целевого устройства без имплементации методов обеспечения сбоеустойчивости;
2) оценить количество строк кода в проекте целевого устройства, защищенного каждым из рассмотренных методов обеспечения сбоеустойчивости;
3) оценить количество строк кода в проекте тестовой системы, защищенной каждым из рассмотренных методов обеспечения сбоеустойчивости, с учетом того, что исходный код тестового конвейера переиспользуется;
4) оценить трудоемкость проекта без использования предложенной в данном исследовании методики выбора способа обеспечения сбоеустойчивости, то есть предполагается, что в процессе разработки была реализована целевая незащищенная система, а затем все версии защищенной целевой системы;
5) оценить трудоемкость проекта с применением методики анализа на основе использования тестовой системы на базе конвейеризированного генератора контрольных кодов, предполагая, что в результате анализа был определен лучший по заданным критериям способ обеспечения сбоеустойчивости, после чего разрабатывается одна защищенная целевая система с применением данного способа обеспечения сбоеустойчивости;
6) сравнить изменение трудоемкости этапа выбора и имплементации способа обеспечения сбоеустойчивости;
7) сравнить общее изменение трудоемкости проекта.

4.2 Сравнение

Во время апробации предложенной методики была проведена серия экспериментов с использованием устройств различного сложности и объема. Далее приведен пример оценки эффективности по критерию трудоемкости применения описанной ранее тестовой системы при выборе метода обеспечения сбоеустойчивости для двух открытых проектов: небольшого

устройства, разработанного автором статьи [12] и ядра процессора с открытым исходным кодом Syntacore SCR1 [13].

4.2.1 Целевая система 1

Первое устройство – блок сбора данных. Устройство (в соответствии с заданным режимом) собирает данные с двух внешних источников данных (данные от одного из них буферизируются), укладывает их в определенной последовательности и передает их на выход. Модель устройства разработана на языке SystemVerilog; в его состав входят конечный автомат, арифметические и логические элементы, а также регистры разрядностью от 1 до 64 бит. Устройство состоит из 10 файлов с RTL кодом, общим объемом 1857 строк.

4.2.2 Целевая система 2

Syntacore SCR1 – 32-битное ядро процессора архитектуры RISC-V с поддержкой архитектуры RV32I, RV32E и наборов команд M и C. Конвейер обладает настраиваемым количеством стадий (от 2 до 4, в эксперименте использована версия минимальная версия с 2 ступенями). Устройство состоит из 11 файлов с RTL кодом, общим объемом 8656 строк.

4.2.3 Разработка защищенных систем

Для реализации защищенных целевых систем было предложено использование одного из способов обеспечения сбоеустойчивости [4].

Было рассмотрено три способа обеспечения сбоеустойчивости:

- защита регистров с использованием трехкратного резервирования с мажорированием и автоматическим восстановлением (порегистровое мажорирование);
- защита регистров с использованием кода Хэмминга (2 ошибки обнаруживаются, 1 исправляется) и автоматическим восстановлением;
- трехкратное резервирование всей системы с мажорированием выходных данных.

При имплементации первых двух способов проводились необходимые доработки функционального описания устройства с вынесением регистров в отдельные инстанции. Для реализации третьего способа был разработан блок верхнего уровня, в котором находятся три инстанции системы и мажорирующий элемент на выходе.

Общий объем доработок для целевого устройств 1 составил:

- способ 1: 124 строки;
- способ 2: 124 строки;
- способ 3: 380 строк.

Суммарный объем кодовой базы (суммарное количество строк исходного проекта и доработок для защищенных систем) составил: 2785 строк.

Новый код (общий объем доработок для имплементации трех рассматриваемых методов обеспечения сбоеустойчивости в целевую систему): 628 строк.

Общий объем доработок для целевого устройства 2 составил:

- способ 1: 453 строки;
- способ 2: 453 строки;
- способ 3: 1110 строк.

Суммарный объем кодовой базы (суммарное количество строк исходного проекта и доработок для защищенных систем) составил: 10672 строк.

Новый код (общий объем доработок для имплементации трех рассматриваемых методов обеспечения сбоеустойчивости в целевую систему): 2062 строк.

Исходя из характеристик целевых систем из имеющихся тестовых систем [9], была выбрана тестовая система на основе конвейеризированного генератора контрольных кодов MD4 [4]. Реализация незащищенной тестовой системы заняла 734 строки. Общий объем доработок при имплементации методов обеспечения сбоеустойчивости составил:

- способ 1: 72 строки;
- способ 2: 72 строки;
- способ 3: 84 строки.

Суммарный объем кодовой базы (суммарное количество строк исходного проекта тестовой системы и доработок для защищенных тестовых систем) составил: 962 строки.

Новый код (общий объем доработок для имплементации трех рассматриваемых методов обеспечения сбоеустойчивости в тестовую систему): 228 строк.

Общий объем кода проекта с применением методики анализа на основе использования тестовой системы на базе конвейеризированного генератора контрольных кодов для целевой системы 1 (суммарное количество строк кода незащищенной целевой системы, трех защищенных тестовых систем и одной защищенной тестовой системы): 352.

Общий объем кода проекта с применением методики анализа на основе использования тестовой системы на базе конвейеризированного генератора контрольных кодов для целевой системы 2: 681.

Необходимо отметить, что, благодаря свойству масштабируемости тестовой системы на основе конвейеризированного генератора контрольного кода изменение размера тестовой системы производится без увеличения объема исходного кода (изменяется значение одного параметра).

Все приведенные в работе системы разработаны на языке SystemVerilog.

4.2.4 Оценка трудоемкости проекта

Полученные оценки трудоемкости приведены в табл. 2.

4.2.5 Сравнение трудоемкости

Для сравнения трудозатрат на этапе выбора и имплементации методов обеспечения сбоеустойчивости вычислим отношение трудоемкости разработки трех защищенных тестовых систем и одной защищенной целевой к трудоемкости разработки трех защищенных версий целевой системы 1:

$$1,5/2,9 = 0,517.$$

В процентном отношении сокращение трудоемкости составляет:

$$(1-(1,5/2,9))\times 100\% = 48,3\%.$$

Для сравнения трудозатрат на разработку всего RTL кода, вычислим отношение трудоемкости всего проекта с имплементацией одного способа обеспечения сбоеустойчивости, выбранного с применением методики анализа на основе использования тестовой системы на базе конвейеризированного генератора контрольных кодов, к трудоемкости всего проекта с реализацией трех защищенных тестовых систем.

В этом случае, суммарная трудоемкости с применением методики анализа способов обеспечения сбоеустойчивости составляет:

$$14,8+1,5=16,3 \text{ человеко-месяца.}$$

Суммарная трудоемкость проекта с реализацией трех защищенных целевых систем:

$$14,8+2,9=17,7 \text{ человеко-месяца.}$$

Соотношение трудозатрат на всю разработку RTL:

16,3/17,7=0,920.

В процентном отношении сокращение трудоемкости составляет:

(1-(16,3 / 17,7)) ×100% = 7,9%.

Табл. 2. Результаты применения различных стратегий.
Table 2. Results for different strategies.

Проект	Объем нового кода	Объем переиспользованного кода	Оценка трудоемкости (человеко-месяцев)
Исходная версия целевой системы 1 (сумма строк RTL кода целевой незащищенной системы)	2785	0	14,8
Исходная версия целевой системы 2 (сумма строк RTL кода целевой незащищенной системы)*	8656	0	48,6*
Тестовые незащищенная и защищенная системы (новый код)	228	734	0,9
Три версии защищенной целевой системы 1 (новый код)	628	2785	2,9
Три версии защищенной целевой системы 2 (новый код)	2016	8656	9,6
Оценка проекта с применением описанной в работе методики (новый код для трех версий защищенной тестовой системы и одна версия защищенной тестовой системы 1)	352	3609	1,5
Оценка проекта с применением описанной в работе методики (новый код для трех версий защищенной тестовой системы и одна версия защищенной тестовой системы 2)	681	8656	2,9

**данные для SRC1 даются на основе оценок по модели COCOMO II.*

Аналогичные расчеты выполняются и для целевой системы 2. Отношение трудоемкости разработки трех защищенных тестовых систем и одной защищенной целевой к трудоемкости разработки трех защищенных версий целевой системы 2:

2,9/9,6 = 0,302.

В процентном отношении сокращение трудоемкости составляет:

(1-(2,9/9,6)) ×100% = 69,8%.

В этом случае, суммарная трудоемкости с применением методики анализа способов обеспечения сбоеустойчивости составляет:

48,6+2,9=51,5 человеко-месяца.

Суммарная трудоемкость проекта с реализацией трех защищенных целевых систем:

14,8+2,9=58,2 человеко-месяца.

Соотношение трудозатрат на всю разработку RTL:

51,5/58,2=0,88.

В процентном отношении сокращение трудоемкости составляет:

(1-(16,3 / 17,7)) ×100% = 11,51%.

Результаты оценки изменения трудоемкость приведены в Табл.3.

Табл. 3. Результаты применения различных стратегий.
Table 3. Results for different strategies.

Проект	Отношение трудозатрат для этапа выбора метода обеспечения сбоеустойчивости	Разница в процентах для этапа выбора метода обеспечения сбоеустойчивости	Отношение трудозатрат для всей разработки RTL	Разница в процентах для всей разработки RTL
Сравнение для проекта целевой системы 1	0,517	48,3%	0,920	7,9%
Сравнение для проекта целевой системы 2 (SCR1)*	0,302	69,8%	0,88	11,51%*

* *оценка трудоемкости разработки полного RTL-кода SCR1 дана на основании модели COCOMO II.*

Выводы

В статье был проведен анализ трудоемкости проекта по критерию трудоемкости для двух процессов разработки: с использованием методики выбора способа обеспечения сбоеустойчивости на основе конвейеризированного контрольного кода и без нее. Сравнение проводилось на основе модели COCOMO II. В результате проведенного сравнения экономия трудозатрат на этапе выбора способа обеспечения сбоеустойчивости при использовании оценки с использованием тестовой системы на основе конвейеризированного генератора контрольного кода составила 48,3%, или 7,9% от всей трудоемкости проекта 1, и 69,8% и 11,51%, соответственно, для проекта 2.

Список литературы / References

[1]. Брехов О.М, Ратников М.О., Сравнительный анализ тестовых систем ПЛИС и их окружения. Труды МАИ, В. 125, 2022, DOI: 10.34759/trd-2022-125-22.

[2]. Райзберг Б.А. Современный экономический словарь. Райзберг Б.А., Лозовский Л.Ш., Стародубцева Е.Б. 6-е изд., перераб. и доп. Москва: ИНФРА-М, 2024. 512 с.

[3]. Boehm B.W. et al. Software Cost Estimation with COCOMO II. Prentice Hall, 2000.

[4]. Santos L.P., Ferreira M. Applying COCOMO II for a DO-178C Safety-Critical Software Effort Estimation. J AerospTechnol Manag, 11: e1819. DOI: 10.5028/jatm.v11.1031.

[5]. Kaur I., Narula G.S., Wason R. et al. Neuro fuzzy – COCOMO II model for software cost estimation. Int. j. inf. tecnol. 10, 181-187, 2018. DOI: 10.1007/s41870-018-0083-6.

[6]. Ren Y., Li Z. Cost Estimation of Software Projects Based on the COCOMOII Model. Atiquzzaman M., Yen N., Xu Z. (eds). Proceedings of the 5th International Conference on Big Data Analytics for Cyber-Physical System in Smart City, vol. 1, BDCPS, 2023. Lecture Notes on Data Engineering and Communications Technologies, vol. 235. Springer, Singapore. DOI:10.1007/978-981-96-0211-7_20.

[7]. Fornaciari W., Salice F., Scarpazza D., Early estimation of the size of VHDL projects. First IEEE/ACM/IFIP International Conference on Hardware. Software Codesign and Systems Synthesis, Newport Beach, CA, USA, 2003, pp. 207-212. DOI: 10.1109/CODES.

[8]. Space Telecommunications Radio System (STRS) application repository design and analysis / Louis M. H. Cleveland, Ohio: National Aeronautics and Space Administration, Glenn Research Center, November 2013.

[9]. Hira A. Calibrating COCOMO® II for Functional Size Metrics: PhD Dissertation, University of Southern California, 2020.

[10]. Brekhov O., Ratnikov M. Pipelined Error-detecting Codes in FPGA Testing. Advances in Electrical and Computer Engineering, 2014, no. 2, vol. 14, pp. 57-62. DOI:10.4316/AECE.2014.02010.

[11]. Brekhov O., Ratnikov M. Using pipelined control code generator as a reference system in FPGA's selection as a hardware platform for fault-tolerance computing system. Journal of Physics: Conference Series, v. 1925, 2021. DOI: 10.1088/1742-6596/1925/1//012038.

[12]. Ratnikov M.O. m.o.ratnikov/disser_sources. Available at: https://gitverse.ru/m.o.ratnikov/disser_sources, accessed 06.11.2025.
[13]. Syntacore GitHub, SCR1 RISC-V Core, Available at: https://github.com/syntacore/scr1, accessed 06.11.2025.

Информация об авторах / Information about authors

Максим Олегович РАТНИКОВ – старший преподаватель кафедры 304 Московского Авиационного Институт, заведующий отделом в АО «ПК «Аквариус». Сфера научных интересов: архитектура вычислительных систем, методы обеспечения сбоеустойчивости вычислительных систем, программируемые логические интегральные схемы, тестирование ПЛИС и систем на базе ПЛИС.

Maksim Olegovitch RATNIKOV – Senior lecturer at Department 304 of the Moscow Aviation Institute, Head of the Department at “PK Aquarius”. Research interests: architecture of computing systems, methods for fault tolerance ensuring of computing systems, field programmable gate array, testing of FPGAs and FPGA-based systems.