ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

ИНФОРМАТИКА

INFORMATICS

ТОМ 22
VOL.

1 | 2025

ОТ РЕДАКЦИИ

В журнале «Информатика» публикуются оригинальные и обзорные статьи, описывающие результаты фундаментальных и прикладных исследований специалистов академического и вузовского профиля в области информатики и информационных технологий.

Основной целью журнала является публикация наиболее значимых новых результатов в указанной области. Приветствуются статьи, описывающие заключительные результаты научных проектов и диссертационных исследований, открывающие новые направления исследований, которые находятся на стыке информатики и других наук.

Журнал рассчитан на широкий круг специалистов в области информатики и информационных технологий.

Основные разделы журнала:

- биоинформатика;
- математическое моделирование;
- защита информации и надежность систем;
- информационные технологии;
- логическое проектирование;
- обработка сигналов, изображений, речи, текста и распознавание образов;
- автоматизация проектирования;
- интеллектуальные системы.

Префикс DOI: 10.37661

Индексирование:

Высшей аттестационной комиссией Республики Беларусь журнал «Информатика» был включен в список научных изданий для опубликования результатов диссертационных исследований.

В декабре 2017 г. включен в базу данных Российского индекса научного цитирования (РИНЦ). С помощью инструментов и сервисов, доступных на платформе eLIBRARY (раздел «Личный кабинет»), можно самостоятельно корректировать список своих публикаций и цитирований в РИНЦ.

В июле 2017 г. включен в базу журналов открытого доступа Directory of Open Access Journals (DOAJ).

С помощью поисковых систем Google Scholar, WorldCat, Соционет можно получить свободный доступ к полному тексту научных публикаций журнала.

Адрес редакции:
ул. Сурганова, 6, к. 305, г. Минск, 220012, Беларусь
Тел. +375 (017) 351 26 22

Editorial address:
Surganova str., 6, of. 305, Minsk, 220012, Belarus
Phone +375 (017) 351 26 22

E-mail: rio@newman.bas-net.by
https://inf.grid.by/jour

THE EDITOR'S NOTE

The journal "Informatics" is a scientific publication in computer sciences and information technologies which reviews the results in basic and applied research of scientists from the universities and scientific centers.

The journal focuses on the most significant and modern papers of research projects results and PhD/DSc thesis in computer sciences.

The journal is edited for the specialists in IT and computer sciences research and application.

The main sections of the journal:

- bioinformatics;
- mathematical modeling;
- information protection and system reliability;
- information technology;
- logical design;
- signal, image, speech, text processing and pattern recognition;
- computer-aided design;
- artificial intelligence methods.

DOI Prefix: 10.37661

Indexation:

the journal "Informatics" is in the list of scientific publications recommended by the Higher Attestation Commission of the Republic of Belarus for scientists to publish the results of PhD/DSc research.

In December 2017 the journal was included in the database of the Russian Science Citation Index (RISC) and provides free access to reviewed electronic scientific paper, improving scientific information traffic and also raising quotation of works of the authors (please use https://elibrary.ru or section for authors https://elibrary.ru_author_tools).

In July 2017 included in the database of open access journals Directory of Open Access Journals (DOAJ).

Using the Google Scholar, WorldCat, Соционет search engine, you can get free access to full text of scientific publications of magazine.

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

ОБЪЕДИНЕННЫЙ ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ
НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК БЕЛАРУСИ

ИНФОРМАТИКА

Informatika

Том 22, № 1, январь-март 2025

Ежеквартальный научный журнал

Издается с января 2004 г.

Учредитель и издатель – государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси» (ОИПИ НАН Беларуси)

ИНФОРМАТИКА

Том 22, № 1, январь-март 2025

Ответственный за выпуск *Мойсейчик Светлана Сергеевна*
Редактор *Гончаренко Галина Борисовна*
Компьютерная верстка *Бутевич Ольга Борисовна*

Сдано в набор 24.02.2025. Подписано в печать 26.03.2025. Формат 60×84 1/8. Бумага офсетная. Гарнитура Таймс. Ризография. Усл. печ. л. 11,6. Уч.-изд. л. 11,4. Тираж 40 экз. Заказ 2.

Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/274 от 04.04.2014. ЛП № 38200000016516 от 18.12.13. Ул. Сурганова, 6, 220012, Минск, Беларусь.

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

THE UNITED INSTITUTE OF INFORMATICS PROBLEMS
OF THE NATIONAL ACADEMY OF SCIENCES OF BELARUS

INFORMATICS

Vol. 22, no. 1, January-March 2025

Published quarterly

Issued since January 2004

Founder and publisher – State Scientific Institution "The United Institute of Informatics Problems of the National Academy of Sciences of Belarus" (UIIP NASB)

INFORMATICS

Vol. 22, no. 1, January-March 2025

Issue Head *Sviatlana S. Maiseichyk*
Editor *Halina B. Hancharenka*
Computer Imposition *Volha B. Butsevich*

Sent for press 24.02.2025. Output 26.03.2025. Format 60×84 1/8. Offset paper. Headset Times. Riesography. Printed sheets 11,6. Publisher’s signatures 11,4. Circulation 40 copies. Order 2.

State Scientific Institution "The United Institute of Informatics Problems of the National Academy of Sciences of Belarus".
Certificate on the state registration of the publisher, manufacturer, distributor of printing editions no. 1/274 dated 04.04.2014. License for the press no. 38200000016516 dated 18.12.13.
6, Surganov Str., 220012, Minsk, Belarus.

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

СОДЕРЖАНИЕ

ISSN 1816-0301 (Print)
ISSN 2617-6963 (Online)

CONTENTS

ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ
LOGICAL DESIGN

УДК 004.33.054
DOI: 10.37661/1816-0301-2025-22-1-7-26

Оригинальная статья
Original Article

Управляемые вероятностные тесты с ограниченным значением расстояния Хэмминга

В. Н. Ярмолик✉, В. В. Петровская, Д. В. Деменковец, В. А. Леванцевич

Белорусский государственный университет
информатики и радиоэлектроники,
ул. П. Бровки, 6, Минск, 220013, Беларусь
✉*E-mail: yarmolik10ru@yahoo.com*

Аннотация

Цели. Решается задача построения управляемых вероятностных тестов с фиксированным минимальным расстоянием Хэмминга. Показывается ограниченность применения классических подходов генерирования тестовых наборов, основанных на перечислении кандидатов в тестовые наборы. С повышением пороговых значений мер различия двоичных тестовых наборов увеличивается вычислительная сложность построения таких тестов. Главной целью настоящей статьи является развитие методов построения тестов на базе исходных шаблонов и правил их расширения до требуемой разрядности.

Методы. На базе расстояния Хэмминга, используемого в теории и практике формирования управляемых вероятностных тестов, рассматриваются новые меры различия для сравнения двух двоичных тестовых наборов. Основой предлагаемых мер различия является формирование множества расстояний Хэмминга для исходных наборов, представляемых в виде последовательностей символов различных алфавитов.

Результаты. Показывается неразличимость пар двоичных тестовых наборов при использовании меры различия, основанной на применении расстояния Хэмминга. В этом случае отличающиеся пары наборов могут иметь совпадающие значения расстояния Хэмминга. Рассматриваются новые меры различия двоичных тестовых последовательностей, которые основаны на их представлении в виде последовательностей, состоящих из символов различных алфавитов. В качестве альтернативы известным решениям предлагается подход, базирующийся на увеличении числа тестовых наборов в тесте при сохранении величины минимального значения расстояния Хэмминга между наборами на приемлемом уровне. Главной особенностью предлагаемого подхода является применение предложенной авторами меры различия, основанной на определении расстояния Хэмминга для тестовых наборов, состоящих из символов различных алфавитов. Показано, что достижение максимального значения расстояния Хэмминга для наборов, представленных большим количеством двоичных символов, обеспечивает такое же значение расстояния для случая, когда символы задаются меньшим числом бит. Это позволяет строить управляемые вероятностные тесты без процедуры перечисления кандидатов в тестовые наборы.

Заключение. Рассмотренные меры различия расширяют возможности генерирования тестовых наборов при формировании управляемых вероятностных тестов. Показывается, что использование различных шаблонов и применяемых к ним правил позволяет строить тесты с фиксированным минимальным расстоянием Хэмминга и требуемой разрядностью тестовых наборов.

Ключевые слова: тестовое диагностирование, управляемые вероятностные тесты, двоичный тестовый набор, мера различия символьных наборов, расстояние Хэмминга, кодочувствительные неисправности

Для цитирования. Управляемые вероятностные тесты с ограниченным значением расстояния Хэмминга / В. Н. Ярмолик, В. В. Петровская, Д. В. Деменковец, В. А. Леванцевич // Информатика. – 2025. − Т. 22, № 1. – С. 7–26. – DOI: 10.37661/1816-0301-2025-22-1-7-26.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 29.10.2024
Подписана в печать | Accepted 10.12.2024
Опубликована | Published 31.03.2025

Controlled random tests with limited Hamming distance

Vyacheslav N. Yarmolik✉, Vita V. Petrovskaya, Denis V. Demenkovets, Vladimer A. Levantsevich

Belarusian State University of Informatics and Radioelectronics,
st. P. Brovki, 6, Minsk, 220013, Belarus
✉*E-mail: yarmolik10ru@yahoo.com*

Abstract

Objectives. The problem of constructing controlled random tests with a fixed minimum Hamming distance is solved. The limitations of classical approaches to generating test patterns based on enumeration of test pattern candidates are shown. With an increase in the threshold values of the difference measures of binary test patterns, the computational complexity of constructing such tests increases. The main goal of this article is to develop methods for constructing tests based on initial templates and rules for expanding them to the required bit size.

Methods. Based on the Hamming distance used in the theory and practice of forming controlled random tests, new measures of difference are considered for comparing two binary test patterns. The basis of the proposed measures of difference is the formation of a set of Hamming distances for the original patterns, represented as sequences of symbols of different alphabets.

Results. The paper demonstrates the indistinguishability of pairs of binary test patterns using a difference measure based on the Hamming distance. In this case, different pairs of patterns may have coinciding Hamming distance values. New measures of difference for binary test sequences based on their representation as sequences consisting of symbols of different alphabets are considered. As an alternative to known solutions, an approach is proposed based on increasing the number of test patterns in a test while maintaining the minimum Hamming distance between patterns at an acceptable level. The main feature of the proposed approach is the use of the difference measure proposed by the authors based on determining the Hamming distance for test patterns consisting of symbols of different alphabets. It is shown that achieving the maximum Hamming distance value for patterns represented by a large number of binary symbols ensures the same distance value for the case when the symbols are specified by a smaller number of bits. This allows one to construct controlled random tests without the procedure of listing candidates for test patterns.

Conclusion. The considered measures of difference expand the possibilities of generating test patterns when forming controlled random tests. It is shown that the use of various templates and rules applied to them allows constructing tests with a fixed minimum Hamming distance and the required bit size of test patterns.

Keywords: test diagnostics, controlled random tests, binary test patterns, character patterns difference measure, Hamming distance, pattern-sensitive faults

For citation. Yarmolik V. N., Petrovskaya V. V., Demenkovets D. V., Levantsevich V. A. *Controlled random tests with limited Hamming distance*. Informatika [*Informatics*], 2025, vol. 22, no. 1, pp. 7−26 (In Russ.). DOI: 10.37661/1816-0301-2025-22-1-7-26.

Conflict of interests. The authors declare of no conflict of interest.

Введение. Тестирование программных приложений является ключевым элементом обеспечения высокого качества программных приложений [1, 2]. Построение тестовых процедур считается фундаментальной задачей при тестировании программного обеспечения. Оценка эффективности различных подходов тестирования породила большой объем исследований, особенно в отношении вероятностного тестирования (random testing) [3, 4]. Отмечается, что несомненным достоинством вероятностного тестирования является автоматизация процесса построения тестов с помощью различных форм вероятностного формирования и отбора тестовых данных и процедур [2, 5]. Этот вид тестирования широко применялся и применяется на практике. Более того, он является основой либо представляет собой составную часть большого числа других разновидностей методов тестирования программных приложений [2, 6].

В ряде работ было показано, что входные данные, вызывающие сбои программных приложений, имеют тенденцию группироваться в форме различного рода кластеров [7–9]. Отмеченная особенность привела к появлению нового семейства методов вероятностного тестирования – так называемого адаптивного вероятностного тестирования (adaptive random testing), которое представляет собой вероятностное тестирование с упорядоченным, чаще всего равномерным распределением тестовых наборов по всей области входных данных [6–9]. В русскоязычных публикациях подобное тестирование часто называют управляемым вероятностным тестированием (controlled random testing) [6]. Данный вид тестирования и его многочисленные модификации основаны на использовании различных стратегий, алгоритмов, оценок и численных характеристик для управляемого формирования очередного случайного тестового набора. Все существующие разновидности управляемого вероятностного тестирования объединяются принципом управляемости генерированием тестовых наборов формируемого теста. Действительно, такие разновидности вероятностных тестов, как упорядоченные вероятностные тесты (orderly random tests), адаптивные вероятностные тесты (adaptive random tests), анти-вероятностные тесты (anti-random tests), эволюционные вероятностные тесты (evolutionary random tests), качественные вероятностные тесты (good random tests), ограниченные вероятностные тесты (restricted random tests), зеркальные вероятностные тесты (mirror random tests), гибридные адаптивные вероятностные тесты (hybrid adaptive random tests), улучшенные адаптивные вероятностные тесты (enhanced adaptive random test), многократное управляемое вероятностное тестирование (multiple controlled random testing) и др., основаны на применении различных характеристик для управляемого генерирования тестовых наборов [10–20].

Большинство известных подходов генерирования адаптивных вероятностных тестов, приведенных выше, основано на использовании расстояния Хэмминга (Hamming distance) в качестве характеристики, определяющей выбор тестовых наборов. Поиск очередного тестового набора из потенциальных кандидатов в тестовые наборы состоит в нахождении такого кандидата, который удовлетворяет заданным критериям, чаще всего определяемым численными значениями используемых характеристик. В случае расстояния Хэмминга таким критерием является само расстояние Хэмминга, пороговое значение которого влияет как на процедуру выбора тестового набора, так и на количество тестовых наборов в тесте. Чем выше значения критериев выбора, в частности расстояния Хэмминга, тем сложнее процедура выбора и заметнее уменьшение длины вероятностного теста [6, 8].

Применение управляемых вероятностных тестов характеризуется большей эффективностью в сравнении с вероятностными, исчерпывающими и псевдоисчерпывающими тестами [1, 2, 6, 21]. Однако необходимость перебора потенциальных кандидатов в тестовые наборы и вычисления для них характеристик существенно увеличивает сложность формирования управляемых вероятностных тестов [6, 8, 9].

Результаты, представленные в данной работе, направлены на решение задачи построения управляемых вероятностных тестов без трудоемких процедур перечисления кандидатов в тесты и вычисления их характеристик. Искомые тесты согласно рассмотренным в статье методам строятся по формальным процедурам на основании шаблонов, представляющих собой тесты с требуемыми характеристиками и малой размерностью. В статье рассматривается случай дво-

ичных управляемых вероятностных тестов, для которых в качестве меры различия используется расстояние Хэмминга и его модификации.

Расстояние Хэмминга и его модификации. В основу методов формирования управляемых вероятностных тестов положена гипотеза, что для двух тестовых наборов, имеющих максимальное различие, количество вновь обнаруживаемых неисправностей (ошибок) вторым набором будет максимальным [6–8]. На основании данной гипотезы показано, что очередной тестовый набор $T_i = t_{i,0}\, t_{i,1}\, t_{i,2} \ldots t_{i,n-1}$, где $t_{i,l} \in \{0, 1\}$, $l \in \{0, 1, 2, \ldots, n-1\}$, управляемого вероятностного теста формируется максимально отличающимся от ранее сгенерированных и включенных в тест наборов $T_0, T_1, T_2, \ldots, T_{i-1}$ [7]. В качестве критерия различия тестового набора T_i по отношению к предыдущим наборам $T_0, T_1, T_2, \ldots, T_{i-1}$ применяется расстояние Хэмминга $HD(T_i, T_j)$ для $j \in \{0, 1, 2, \ldots, i-1\}$, определяемое согласно соотношению

$$HD(T_i, T_j) = \sum_{l=0}^{n-1} \mathrm{I}(t_{i,l} \neq t_{j,l}). \quad (1)$$

Выражение $\mathrm{I}(t_{i,l} \neq t_{j,l})$ представляет собой индикаторную функцию, равную единице при $t_{i,l} \neq t_{j,l}$ и нулю в противном случае [22, 23]. Минимальное значение $\min HD(T_i, T_j) = 0$ при совпадении всех символов последовательностей T_i и T_j, а максимальное $\max HD(T_i, T_j) = n$ при их несовпадении.

В ряде работ отмечалось, что расстояние Хэмминга (1) как мера различия малоэффективна, так как она позволяет различать лишь полностью совпадающие последовательности при $HD(T_i, T_j) = 0$ и все остальные несовпадающие [24–27]. Аргументом для подтверждения неразличимости несовпадающих последовательностей являются наборы двоичных символов T_i и $T_j = \overline{T_i}$, значение расстояния Хэмминга $HD(T_i, \overline{T_i})$ для которых всегда неизменно и равняется n несмотря на то, что T_i представляет собой произвольный двоичный набор из n символов.

Для более полной оценки различия двоичных наборов в работе [25] была определена новая мера различия для случая, когда $n = 2^m$, где m – натуральное число либо нуль. Данная мера различия двоичных тестовых наборов T_i и T_j состоит из множества численных характеристик HD_0, HD_1, ..., HD_v, …, HD_m, представляющих собой расстояния Хэмминга $HD_v[T_i(2^v), T_j(2^v)]$ для указанных наборов $T_i(2^v)$ и $T_j(2^v)$, состоящих из символов, которые заданы их 2^v последовательными битами, где $v \in \{0, 1, 2, \ldots, m\}$.

Требование к размерности $n = 2^m$ бинарных тестовых наборов T_i не всегда выполняется на практике. Соответственно, для $n \neq 2^m$ при отображении исходного набора T_i в последовательности $T_i(1)$, $T_i(2)$, $T_i(4)$, … может отсутствовать необходимое количество бит, равное 2^v для последнего символа последовательности $T_i(2^v)$, $v \in \{0, 1, 2, \ldots, m\}$. Например, поскольку для тестового набора $T_i = 0100101_{(2)}$ n равно семи, его можно представить в виде последовательностей $T_i(1)$, $T_i(2)$, $T_i(4)$ и $T_i(8)$. Однако в трех случаях, а именно $T_i(2)$, $T_i(4)$ и $T_i(8)$, для последнего символа соответствующего алфавита отсутствует необходимое количество бит, а именно один бит.

Очевидным решением для устранения данного ограничения является циклическая интерпретация исходного набора $T_i = t_{i,0}\, t_{i,1}\, t_{i,2} \ldots t_{i,n-1}$ [25]. Такая интерпретация предполагает, что следующим битом тестового набора T_i после последнего бита $t_{i,n-1}$ является его первый бит $t_{i,0}$. Таким образом, используются первые разряды набора для получения необходимого количества бит для последнего символа $T_i(2^v)$. В случае тестового набора $T_i = 0100101_{(2)}$ циклическая интерпретация позволяет получить $T_i(2) = 0100101\underline{\mathbf{0}}_{(2)} = 1022_{(4)}$; $T_i(4) = 0100101\underline{\mathbf{0}}_{(2)} = 4\mathrm{A}_{(16)}$; $T_i(8) = 0110001\underline{\mathbf{0}}_{(2)} = \mathrm{b}_{(256)}$.

Снятие ограничения на размерность n двоичного набора T_i путем его расширения до требуемого числа разрядов позволяет увеличить количество алфавитов для его отображений. Очевидно, что с учетом возможности расширения исходного двоичного тестового набора до требуемого числа бит число алфавитов может быть увеличено до n. Эти алфавиты состоят из символов, задаваемых одним битом, двумя, тремя, четырьмя битами и т. д. вплоть до алфавита, в котором каждый символ определяется n последовательными битами [25, 27]. Рассматривая пример исходного набора $T_i = 01010_{(2)}$ ($n = 5$) и его циклических расширений, представим его в виде после-

довательностей, полученных для $n = 5$ алфавитов. Соответственно имеем: $T_i = T_i(1) = 01010_{(2)}$; $T_i(2) = 01010\underline{\mathbf{0}}_{(2)} = 110_{(4)}$; $T_i(3) = 01010\underline{\mathbf{0}}_{(2)} = 24_{(8)}$; $T_i(4) = 01010\underline{\mathbf{010}}_{(2)} = 52_{(16)}$; $T_i(5) = 01010_{(2)} = A_{(32)}$.

Кроме циклического расширения разрядности тестовых наборов может быть рассмотрено их расширение значениями, которые либо генерируются как случайные бинарные величины, либо формируются стандартным образом. Эффект расширения тестовых наборов оказывает влияние только на величины численных характеристик HD_v, для которых исходные тестовые наборы представляются в алфавитах, состоящих из большого числа двоичных символов [25].

Рассмотрим представление исходного тестового набора T_i с произвольным количеством n двоичных символов путем расширения последнего символа набора стандартным образом, например нулевыми значениями двоичных символов. Для этого приведем пример тестового набора $T_i = 01010$, который может быть представлен в пяти различных системах счисления, использующих свой алфавит. В случае набора $T_i = 01010$ применяются алфавиты, состоящие из 2, 4, 8, 16 и 32 двоичных символов. Чтобы избежать конфликтных ситуаций, связанных с отсутствием полного набора символов (их графических изображений) для алфавитов, которые включают большое число двоичных символов, каждый символ во всех системах счисления будем представлять в двоичном коде, а сами символы разделять пробелами. Таким образом, тестовый набор $T_i = 01010$ может быть представлен в пяти различных системах счисления: $T_i(1) = 0\ 1\ 0\ 1\ 0_{(2)}$; $T_i(2) = 01\ 01\ 0\underline{\mathbf{0}}_{(4)}$; $T_i(3) = 010\ 10\underline{\mathbf{0}}_{(8)}$; $T_i(4) = 0101\ 0\underline{\mathbf{000}}_{(16)}$; $T_i(5) = 01010_{(32)}$. Используя приведенный пример, дадим определение двоичного n-разрядного тестового набора T_i как набора, представленного в другой, не двоичной, системе счисления.

Определение 1. Тестовый набор T_i, включающий n двоичных символов, можно интерпретировать в системе счисления с 2^r символами как набор $T_i(r)$, $r \in \{1, 2, 3, \ldots, n\}$, состоящий из $\lceil n/r \rceil$ символов, каждый из которых представляется r последовательными битами набора T_i. Для этого T_i расширяется до размерности $\lceil n/r \rceil \cdot r$ путем добавления $\lceil n/r \rceil \cdot r - n$ нулей.

Например, $T_i = 0101001$ с $n = 7$ можно представить в восьмеричной (2^3) системе счисления $\lceil n/r \rceil = \lceil 7/3 \rceil = 3$ символами как $T_i(3) = 010\ 100\ 1\underline{\mathbf{00}}_{(8)} = 244_{(8)}$. Для получения этого результата необходимо добавить $\lceil n/r \rceil \cdot r - n = \lceil 7/3 \rceil \cdot 3 - 7 = 2$ нуля. Определение 1 и приведенные выше примеры интерпретации двоичного набора T_i позволяют рассматривать бинарные тестовые наборы в различных системах счисления. Используя пример представления тестовых наборов $T_i = 01010$ и $T_j = 11001$ в различных системах счисления, приведем результаты определения (табл. 1) для каждой из их интерпретаций расстояния Хэмминга $HD(T_i, T_j)$ (1).

Таблица 1
Пример вычисления расстояния Хэмминга $HD(T_i, T_j)$

Table 1
Example of calculating the Hamming distance $HD(T_i, T_j)$

T_i	$T_i(1) = 0\ 1\ 0\ 1\ 0$	$T_i(2) = 01\ 01\ 0\underline{\mathbf{0}}$	$T_i(3) = 010\ 10\underline{\mathbf{0}}$	$T_i(4) = 0101\ 0\underline{\mathbf{000}}$	$T_i(5) = 01010$
T_j	$T_j(1) = 1\ 1\ 0\ 0\ 1$	$T_j(2) = 11\ 00\ 1\underline{\mathbf{0}}$	$T_j(3) = 110\ 01\underline{\mathbf{0}}$	$T_j(4) = 1100\ 1\underline{\mathbf{000}}$	$T_j(5) = 11001$
$HD(T_i, T_j)$	3	3	2	2	1

Приведенный выше пример определения расстояния Хэмминга показывает возможность получения на основе равенства (1) нескольких числовых оценок соотношения тестовых двоичных наборов T_i и T_j. Основываясь на этом примере, определим новую меру различия между бинарными тестовыми наборами T_i и T_j, состоящую из нескольких числовых характеристик, являющихся расстояниями Хэмминга.

Определение 2. Мера различия $MD(T_i, T_j)$ между тестовыми наборами $T_i = t_{i,0}\ t_{i,1}\ t_{i,2} \ldots t_{i,n-1}$ и $T_j = t_{j,0}\ t_{j,1}\ t_{j,2} \ldots t_{j,n-1}$, где $t_{i,l}, t_{j,l} \in \{0, 1\}$, $l \in \{0, 1, 2, \ldots, n-1\}$, состоит из n числовых компонент, представленных значениями расстояния Хэмминга $HD_1 = HD[T_i(1), T_j(1)]$, $HD_2 = HD[T_i(2), T_j(2)]$, $HD_3 = HD[T_i(3), T_j(3)]$, …, $HD_n = HD[T_i(n), T_j(n)]$ между наборами $T_i(r)$ и $T_j(r)$, $r \in \{1, 2, 3, \ldots, n\}$.

Анализируемые символы $t_{i,l}$ и $t_{j,l}$ тестовых наборов $T_i(r)$ и $T_j(r)$ согласно определению 1 представлены $r \in \{1, 2, 3, \ldots, n\}$ двоичными битами. Соответственно, применяя (1), формируются числовые значения компонент HD_1, HD_2, HD_3, …, HD_n меры различия $MD(T_i, T_j)$. Подобная ме-

ра различия рассмотрена в работе [25] для случая, когда $n = 2^m$, где m – натуральное число либо нуль, который исключает возможность представления тестовых наборов символами, состоящими из трех, пяти, шести, семи и так далее бит.

В табл. 2 приведены примеры вычисления $MD(T_i, T_j)$ для различных пар тестовых наборов T_i и T_j для случая, когда $n = 5$.

Таблица 2
Примеры вычисления расстояния Хэмминга $HD(T_i, T_j)$

Table 2
Examples of calculating the Hamming distance $HD(T_i, T_j)$

T_i	$T_i(1)$ = 0 1 0 1 0	$T_i(2)$ = 01 01 00	$T_i(3)$ = 010 100	$T_i(4)$ = 0101 0000	$T_i(5)$ = 01010
T_j	$T_j(1)$ = 1 0 0 0 0	$T_j(2)$ = 10 00 00	$T_j(3)$ = 100 000	$T_j(4)$ = 1000 0000	$T_j(5)$ = 10000
$HD(T_i, T_j)$	3	2	2	1	1
T_i	$T_i(1)$ = 0 1 0 1 0	$T_i(2)$ = 01 01 00	$T_i(3)$ = 010 100	$T_i(4)$ = 0101 0000	$T_i(5)$ = 01010
T_j	$T_j(1)$ = 1 0 0 1 1	$T_j(2)$ = 10 01 10	$T_j(3)$ = 100 110	$T_j(4)$ = 1001 1000	$T_j(5)$ = 10011
$HD(T_i, T_j)$	3	2	2	2	1

Отметим, что во всех трех примерах, приведенных в табл. 1 и 2, в качестве тестового набора T_i использовался один и тот же набор T_i = 01010, а для определения значений меры $MD(T_i, T_j)$ были выбраны три различных набора T_j = 11001, T_j = 10000 и T_j = 10011. Соответственно, для трех случаев, представленных в табл. 1 и 2, мера различия $MD(T_i, T_j)$ принимает следующие значения: MD(01010, 11001) = 3, 3, 2, 2, 1; MD(01010, 10000) = 3, 2, 2, 1, 1; MD(01010, 10011) = = 3, 2, 2, 2, 1.

Примеры, приведенные в табл. 1 и 2, показывают неразличимость всех трех тестовых наборов T_j относительно набора T_i = 01010 при использовании классической меры различия – расстояния Хэмминга, поскольку во всех трех случаях $HD(T_i, T_j) = HD_1 = 3$. В то же время применение новой меры различия (см. определение 2) показывает неодинаковую степень различия наборов T_j от T_i, выраженную в отличающихся значениях компонент HD_2, HD_3 и HD_4 меры $MD(T_i, T_j)$.

Мера различия $MD(T_i, T_j)$ для двоичных тестовых наборов T_i и T_j обладает следующими очевидными свойствами.

Свойство 1. *Минимальное числовое значение компонент HD_1, HD_2, HD_3, …, HD_n меры различия $MD(T_i, T_j)$ равно 0, т. е.* $\min HD_1 = \min HD_2 = \min HD_3 = \ldots = \min HD_n = 0$.

Все компоненты HD_1, HD_2, HD_3, …, HD_n меры $MD(T_i, T_j)$ равняются нулю при совпадении тестовых наборов, т. е. при $T_i = T_j$ [25].

Свойство 2. *Если одна компонента HD_r, $r \in \{1, 2, 3, \ldots, n\}$, равна нулю, то все остальные также равны нулю, и наоборот, если одна компонента HD_r больше нуля, то все остальные также больше нуля.*

Свойство 3. *Максимальные значения компонент HD_1, HD_2, HD_3, …, HD_n меры различия $MD(T_i, T_j)$ определяются числом $\lceil n/r \rceil$, $r \in \{1, 2, 3, \ldots, n\}$, символов в сравниваемых наборах $T_i(r)$ и $T_j(r)$ и, соответственно, принимают следующие значения:* $\max HD_1 = n$; $\max HD_2 = \lceil n/2 \rceil$; $\max HD_3 = \lceil n/3 \rceil$; …; $\max HD_{\lceil n/2 \rceil - 1} = \lceil n/(\lceil n/2 \rceil - 1) \rceil = 3$; $\max HD_{\lceil n/2 \rceil} = \max HD_{\lceil n/2 \rceil + 1} = \max HD_{\lceil n/2 \rceil + 2} = \ldots = \max HD_{n-1} = 2$; $\max HD_n = 1$.

Максимальное различие между тестовыми наборами T_i и T_j согласно новой мере различия $MD(T_i, T_j)$ достигается для случая, когда T_j является инверсным набором по отношению к T_i. В этом случае все компоненты HD_1, HD_2, HD_3, …, HD_n меры $MD(T_i, T_j)$ принимают максимальные значения. В случае когда T_i = 01010, а инверсный набор T_j равняется $\overline{T_i}$ = 10101, соответствующие компоненты имеют следующие значения: $HD_1 = \max HD_1 = n = 5$; $HD_2 = \max HD_2 = \lceil n/2 \rceil = \lceil 5/2 \rceil = 3$; $HD_3 = \max HD_3 = \lceil n/3 \rceil = \lceil 5/3 \rceil = 2$; $HD_4 = \max HD_4 = \lceil n/4 \rceil = \lceil 5/4 \rceil = 2$; $HD_5 = \max HD_5 = 1$.

Свойство 4. *Числовые значения компонент HD_1, HD_2, HD_3, …, HD_n меры различия $MD(T_i, T_j)$ связаны следующими соотношениями:* $HD_1 \geq HD_2 \geq HD_3 \geq \ldots \geq HD_n$.

Выполнение свойства 4 объясняется тем, что при вычислении HD_{r+1} число символов, входящих в наборы $T_i(r+1)$ и $T_j(r+1)$, меньше или равно числу символов в наборах $T_i(r)$ и $T_j(r)$, поэтому $HD_r[T_i(r), T_j(r)] \geq HD_{r+1}[T_i(r+1), T_j(r+1)]$.

Как отмечено в работах [6, 8–10], основная идея построения управляемых вероятностных тестов заключается в том, что очередной тестовый набор T_i формируется максимально отличным (удаленным) от ранее сформированных наборов $T_0, T_1, T_2, \ldots, T_{i-1}$ в соответствии с заранее выбранными мерами различия. Для этого на каждом шаге формирования очередного тестового набора он выбирается из множества кандидатов в тестовые наборы [6, 8–10]. Основной операцией процедуры выбора одного из кандидатов является определение численного значения используемой меры различия между двумя наборами T_i и T_j, первый из которых является одним из тестовых наборов, а второй – одним из кандидатов в тестовый набор. В результате в качестве очередного тестового набора выбирается тот кандидат в тестовый набор, для которого значение меры различия принимает максимальное значение. Поясним классическую процедуру синтеза управляемых вероятностных тестов на простейшем примере.

Предполагая, что первым тестовым набором теста является набор $T_0 = 01010$, случайным образом генерируем, например, три кандидата в тестовый набор T_1, а именно 11001, 10000 и 10011. Затем для каждого кандидата в тестовый набор T_1 согласно определению 2 вычисляется значение меры различия относительно тестового набора T_0. Как видно из табл. 1 и 2, значение HD_1 во всех трех случаях равно трем. Классическая методика построения управляемых вероятностных тестов предполагает использование любого из трех кандидатов 11001, 10000 и 10011 в качестве тестового набора T_1. При формировании последующих тестовых наборов вычисляются значения мер различия между наборами, входящими в тест, и кандидатами в тестовые наборы [6–10].

В случае получения максимального значения HD_1 для нескольких кандидатов в тестовые наборы введенная авторами новая мера различия $MD(T_i, T_j)$ (см. определение 2) позволяет более полно учесть различия кандидатов в тестовые наборы T_j относительно набора T_i. Для этого необходимо проанализировать значения следующей компоненты HD_2 предложенной авторами меры различия. Как видно из рассматриваемого ранее примера (см. табл. 1 и 2), максимальное значение $HD_2 = 3$ достигается для набора $T_j = 11001$, который в дальнейшем может быть использован в качестве тестового набора T_1. Основываясь на приведенном выше примере и руководствуясь классической стратегией генерирования управляемых вероятностных тестов, сформулируем утверждение, которым необходимо руководствоваться при использовании меры различия $MD(T_i, T_j)$.

Утверждение 1. *В качестве максимально отличающегося набора T_j по отношению к набору T_i принимается такой кандидат в наборы T_j, который только один из всего множества кандидатов в T_j имеет максимальное значение компоненты HD_r меры различия $MD(T_i, T_j) = HD_1, HD_2, HD_3, \ldots, HD_n$ для минимального $r \in \{1, 2, 3, \ldots, n\}$, иначе случайным образом выбирается один из кандидатов, который принадлежит подмножеству кандидатов с максимальным значением компоненты HD_r меры различия $MD(T_i, T_j)$.*

Предложенная мера различия $MD(T_i, T_j)$ показывает свою эффективность при построении управляемых вероятностных тестов. Она позволяет выбирать оптимальный тестовый набор T_j для управляемого вероятностного теста из множества кандидатов в тестовый набор, объединенных равенством их классического расстояния Хэмминга с тестовым набором T_i, ранее включенным в тест. В то же время использование данной меры различия сопряжено с теми же недостатками, что и у классических подходов, требующих значительных вычислительных затрат. Прежде всего это касается необходимости определения различий, т. е. вычисления $MD(T_i, T_j)$, между тестовыми наборами T_j как кандидатами в тест и тестовыми наборами T_i, ранее включенными в тест.

Анализ и синтез управляемых вероятностных тестов с фиксированным минимальным расстоянием Хэмминга. В общем случае управляемый вероятностный тест характеризуется минимальным значением расстояния Хэмминга $\min HD(T_i, T_j)$, которое, по сути, является его основной характеристикой. Этот параметр соответствует следующему определению.

Определение 3. Значение $\min HD(T_i, T_j)$ равняется минимальному расстоянию Хэмминга между двумя произвольными тестовыми наборами T_i и T_j, $i \neq j \in \{0, 1, 2, \ldots, q-1\}$, из множества наборов теста $T = \{T_0, T_1, T_2, \ldots, T_{q-1}\}$.

В терминах теории помехоустойчивого кодирования характеристику $\min HD(T_i, T_j)$ можно рассматривать как кодовое расстояние h кода T, которое равняется наименьшему расстоянию Хэмминга между различными парами кодовых слов $T_0, T_1, T_2, \ldots, T_{q-1}$. Поэтому, исходя из основополагающих положений теории кодирования, были сформулированы полезные выводы, которые необходимо учитывать при генерировании управляемых вероятностных тестов [6, 28–30]. Отмечено, что особенностью управляемых вероятностных тестов является ограниченность их длины, которая определяется значением $\min HD(T_i, T_j)$, используемым как критерий включения тестового набора в тест. Чем больше $h = \min HD(T_i, T_j)$, тем меньше количество q наборов, удовлетворяющих такому критерию. Это следует из предельной оценки Хэмминга (Hamming bound) [6, 23, 29, 30]. Для случая двоичного алфавита, представляя нечетные $h = \min HD(T_i, T_j)$ как $h = 2z + 1$, а четные как $h = 2z + 2$, где z – натуральное число либо нуль, данная оценка задается неравенством

$$q \leq 2^n / \sum_{k=0}^{z} \binom{n}{k}. \tag{2}$$

Например, в случае когда $n = 8$ и $h = \min HD(T_i, T_j) = 3 = 2 \cdot 1 + 1$ согласно (2), оказывается возможным построение управляемого вероятностного теста, состоящего из

$$q \leq 2^8 / \sum_{k=0}^{1} \binom{8}{k} = 2^8 / (1+8) = 28,444...$$

тестовых наборов со значением расстояния Хэмминга $HD(T_i, T_j)$, большим или равным трем. Увеличение расстояния Хэмминга $\min HD(T_i, T_j)$, например, до значения семь, уменьшает оценку q до величины два, откуда следует, что управляемый вероятностный тест для $n = 8$ и $h = \min HD(T_i, T_j) = 7$ будет состоять не более чем из двух наборов $T = \{T_0, T_1\}$. Очевидным образом размерность q управляемого вероятностного теста сокращается с уменьшением разрядности наборов n. Для того же случая $h = \min HD(T_i, T_j) = 3$, но вдвое меньшей разрядности $n = 4$, оценка q согласно (2) уменьшается до величины три.

Для синтеза управляемых вероятностных тестов с малым количеством тестовых наборов q в работе [29] рассмотрены управляемые вероятностные тесты $MMHD(q)$ (Maximum Minimum Hamming Distance) с максимальным минимальным расстоянием Хэмминга $\min HD(T_i, T_j) > n/2$. Показано, что теорема Плоткина позволяет определить максимально возможное количество q кодовых слов в двоичном коде длины n для максимального кодового расстояния h, а граница Плоткина дает верхний предел этого количества [28].

Построение управляемых вероятностных тестов $MMHD(q) = \{T_0, T_1, T_2, \ldots, T_{q-1}\}$ с малым количеством тестовых наборов q заключается в обеспечении максимально возможного значения $h = \min HD(T_i, T_j) > n/2$ для фиксированных значений q [28]. Граница Плоткина позволяет получить оценки максимального значения минимального расстояния $\max\_\min HD(T_i, T_j)$ для заданных значений q. Как показано в работах [6, 28, 29] для случая $q = 2$ и произвольной разрядности тестовых наборов n, значение $\max\_\min HD(T_i, T_j)$ равняется n, а сам тест включает два взаимно инверсных набора. Для случая $q = 3$ в соответствии с теоремой Плоткина имеем следующую оценку: $\max\_\min HD(T_i, T_j) \leq 3n/4$. Однако для тестов $MMHD(3)$ и $MMHD(4)$, как показано в статье [29], $\max\_\min\{MMHD(3)\} = \max\_\min\{MMHD(4)\} = 2n/3$.

С использованием расстояния Хэмминга $HD(T_i, T_j)$ для тестовых наборов T_i и T_j и их декартова расстояния $CD(T_i, T_j)$ в работе [18] рассмотрен метод синтеза оптимальных управляемых

вероятностных тестов (Optimal Controlled Random Tests, OCRT). Подобные тесты характеризуются тем, что для них $h = \min HD(T_i, T_j) \geq n/2$. В общем случае количество наборов *OCRT* определяется как $q = 2(\lceil \log_2 n \rceil + 1)$, а алгоритм для формирования тестовых наборов представлен в работах [6, 18]. Для случая когда $n = 2^m$, где m – натуральное число, количество q наборов $OCRT = \{T_0, T_1, T_2, \ldots, T_{q-1}\}$ равняется $2(m + 1)$. Например, для $n = 4$ количество тестовых наборов *OCRT* $q = 6$, а для $n = 8$, соответственно, $q = 8$. Пример теста $MMHD(4)$ с $HD(T_i, T_j) = 2$ для $n = 3$, представленный в работе [29], и пример теста *OCRT* для $n = 4$ [30] показаны на рис. 1.

Тесты $MMHD(q)$ при их рассмотрении представлялись для минимальной разрядности в качестве шаблонов, на основании которых они расширялись до требуемой разрядности [28].

Общим недостатком известных подходов синтеза управляемых вероятностных тестов являются ограничения на структуру и размерность исходных тестовых шаблонов, на основании которых строится тест с требуемой разрядностью n тестовых наборов $T_i = t_{i,0}\, t_{i,1} \ldots t_{i,n-1}$.

MMHD(4)	
T_0	0 0 0
T_1	0 1 1
T_2	1 1 0
T_3	1 0 1

OCRT	
T_0	0 0 0 0
T_1	1 1 1 1
T_2	0 0 1 1
T_3	1 1 0 0
T_4	0 1 0 1
T_5	1 0 1 0

Рис. 1. Примеры теста $MMHD(4)$ и теста *OCRT* для $n = 4$
Fig. 1. MMHD(4) *and OCRT examples for n = 4*

Под тестовыми шаблонами в дальнейшем будем понимать управляемый вероятностный тест $CRT(q, h, n)$ с фиксированным количеством тестовых наборов q и заданным минимальным значением расстояния Хэмминга $h = \min HD(T_i, T_j)$, построенный для минимальной разрядности n тестовых наборов.

Расстояние Хэмминга определяется количеством несовпадающих бит h относительно величины разрядности n тестовых наборов и принимает значения $h \leq n$. С помощью подобных шаблонов с заданными характеристиками q и h строится управляемый вероятностный тест для требуемой разрядности n тестовых наборов, в котором сохраняется относительное h/n значение расстояния Хэмминга. На основании исходного шаблона $CRT(q, h, n)$ либо произвольного управляемого вероятностного теста оказывается возможным построение их семейства при использовании правил преобразования [22, 23, 29]. Эти правила основаны на свойствах двоичных кодов, исследуемых в теории помехоустойчивого кодирования, и позволяют модифицировать исходный шаблон, сохраняя при этом значения и соотношения его характеристик q, h и n.

Правило 1. Результатом перестановки местами в тесте $CRT(q, h, n)$ тестовых наборов $T_i = t_{i,0}\, t_{i,1}\, t_{i,2} \ldots t_{i,n-1}$ и $T_j = t_{j,0}\, t_{j,1}\, t_{j,2} \ldots t_{j,n-1}$, $i \neq j \in \{0, 1, 2, \ldots, q-1\}$, является тест $CRT(q, h, n)$.

Правило 2. При перестановке пары одноименных бит во всех q тестовых наборах $T_0, T_1, T_2, \ldots, T_{q-1}$ теста $CRT(q, h, n)$ формируется также тест $CRT(q, h, n)$.

Правило 3. Результатом инвертирования произвольного множества одноименных бит во всех q тестовых наборах $T_0, T_1, T_2, \ldots, T_{q-1}$ теста $CRT(q, h, n)$ также является тест $CRT(q, h, n)$.

Правило 3 вытекает из свойства индикаторной функции $I(t_{i,l} \neq t_{j,l})$, используемой в выражении (1), которое описывается равенством $I(t_{i,l} \neq t_{j,l}) = I(\overline{t_{i,l}} \neq \overline{t_{j,l}})$.

Как уже отмечалось, приведенные выше правила применимы для произвольного $CRT(q, h, n)$ и позволяют строить семейства подобных шаблонов и тестов. Результат применения данных правил для теста $MMHD(4)$, который можно рассматривать как шаблон $CRT_0(4, 2, 3)$, представлен в табл. 3.

Таблица 3
Применение правил преобразования для *MMHD*(4)
Table 3
Transformation rules application for MMHD(4)

Шаблон $CRT_0(4, 2, 3)$ *Sample* $CRT_0(4, 2, 3)$	Правило 1 $CRT_1(4, 2, 3)$ *Rule 1* $CRT_1(4, 2, 3)$	Правило 2 $CRT_2(4, 2, 3)$ *Rule 2* $CRT_2(4, 2, 3)$	Правило 3 $CRT_3(4, 2, 3)$ *Rule 3* $CRT_3(4, 2, 3)$
$T_i = t_{i,0}\, t_{i,1}\, t_{i,2}$	$T_i = t_{i,0}\, t_{i,1}\, t_{i,2}$	$T_i = t_{i,1}\, t_{i,0}\, t_{i,2}$	$T_i = t_{i,0}\, t_{i,1}\, \overline{t_{i,2}}$
$T_0 = 0\ \ 0\ \ 0$	$T_3 = 1\ \ 0\ \ 1$	$T_0 = 0\ \ 0\ \ 0$	$0\ \ 0\ \ 1$
$T_1 = 0\ \ 1\ \ 1$	$T_1 = 0\ \ 1\ \ 1$	$T_3 = 1\ \ 0\ \ 1$	$0\ \ 1\ \ 0$
$T_2 = 1\ \ 1\ \ 0$	$T_2 = 1\ \ 1\ \ 0$	$T_2 = 1\ \ 1\ \ 0$	$1\ \ 1\ \ 1$
$T_3 = 1\ \ 0\ \ 1$	$T_0 = 0\ \ 0\ \ 0$	$T_1 = 0\ \ 1\ \ 1$	$1\ \ 0\ \ 0$

Применив правило 1 к исходному тесту *MMHD*(4), обозначенному в табл. 3 как шаблон $CRT_0(4, 2, 3)$, путем замены местами тестовых наборов T_0 и T_3, получим новый тест $CRT_1(4, 2, 3)$, который в дальнейшем может использоваться как шаблон или тест. Правило 2 также позволило сформировать новый шаблон $CRT_2(4, 2, 3)$ путем перестановки местами нулевого и первого бита во всех четырех тестовых наборах (табл. 3). Отметим, что в силу специфики структуры теста *MMHD*(4) результат применения правила 2 эквивалентен применению правила 1 при замене местами тестовых наборов T_1 и T_3. Использование правила 3 также позволило построить новый шаблон $CRT_3(4, 2, 3)$, который, как и исходный тест *MMHD*(4), и результаты применения к нему правил 1 и 2 (табл. 3), можно интерпретировать как шаблон $CRT(q, h, n)$ с соответствующими параметрами q, h и n. В случае правила 3 новый шаблон $CRT_3(4, 2, 3)$ содержит тестовые наборы, которые не присутствуют в исходном шаблоне. Необходимо отметить, что на основании исходного теста (шаблона) и одновременного применения нескольких правил также возможно получение нового шаблона.

Управляемый вероятностный тест *MMHD*(4), как и полученные согласно правилам 1–3 его модификации, можно рассматривать в качестве шаблонов $CRT(q, h, n) = CRT(4, 2, 3)$ для построения тестов произвольной разрядности n с фиксированными значениями количества наборов $q = 4$ и минимального относительного значения расстояния Хэмминга $h/n = 2/3$. В общем случае построение тестов на основании шаблонов для заданной разрядности n основано на применении следующих правил. Во всех последующих правилах, так же как и в трех предыдущих, в качестве шаблона может принимать участие любой управляемый вероятностный тест.

Правило 4. Правило повторения заключается в построении теста $CRT(q, w{\cdot}h, w{\cdot}n)$ путем w-кратного повторения исходного шаблона $CRT(q, h, n)$.

В качестве примера применения данного правила можно рассмотреть управляемый вероятностный тест $CRT(4, 4{\cdot}2, 4{\cdot}3)$, полученный в результате четырехкратного повторения шаблона $CRT_0(4, 2, 3)$ (рис. 2).

$$\begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{array}$$

Рис. 2. Управляемый вероятностный тест *CRT*(4, 8, 12)
Fig. 2. Controlled random test CRT(4, 8, 12)

Содержательно близким по своей сути к правилу повторения является правило объединения, которое можно рассматривать как расширение правила повторения.

Правило 5. Правило объединения заключается в построении теста $CRT(q, h, n)$ путем повторения различных шаблонов с одинаковым числом тестовых наборов q, сумма количества бит

которых равняется n. При этом значения расстояний Хэмминга в используемых шаблонах могут быть разными, а их сумма определяет величину h.

В качестве примера используем шаблоны $CRT_0(4, 2, 3)$, $CRT_2(4, 2, 3)$, $CRT_3(4, 2, 3)$, а также шаблон $CRT(4, 2, 4)$. В результате применения правила 5 получим $CRT(4, 8, 13)$ (рис. 3).

0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	1	0	1	0	1	0	0	1	0	1
1	1	0	1	1	0	1	1	1	1	0	1	0
1	0	1	0	1	1	1	0	0	1	1	1	1

Рис. 3. Управляемый вероятностный тест $CRT(4, 8, 13)$
Fig. 3. Controlled random test $CRT(4, 8, 13)$

На рис. 3 видно, что результирующий управляемый вероятностный тест $CRT(4, 8, 13)$ имеет разрядность $n = 13$, равную сумме разрядностей используемых шаблонов. Как и значение $h = 8$, он определяется суммой расстояний Хэмминга используемых четырех шаблонов. Отметим, что при использовании правила 5 последовательность применения шаблонов не имеет принципиального значения, так как на основании теста $CRT(4, 8, 13)$ можно получить семейство подобных тестов с помощью ранее определенных правил 1–3.

Правило 6. Результатом масштабирования (увеличения) в w раз шаблона $CRT(q, h, n)$ является управляемый вероятностный тест $CRT(q, w{\cdot}h, w{\cdot}n)$, состоящий из q тестовых наборов $T_i = (t_{i,0})^w (t_{i,1})^w (t_{i,2})^w \ldots (t_{i,n-1})^w$, где $T_i = t_{i,0}\, t_{i,1}\, t_{i,2} \ldots t_{i,n-1}$ – тестовый набор исходного шаблона $CRT(q, h, n)$.

Иллюстрацией применения данного правила может быть пример получения теста $CRT(4, 3{\cdot}2, 3{\cdot}3)$ на основании шаблона $CRT_0(4, 2, 3)$ для $w = 3$ (рис. 4).

0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	0	0	0
1	1	1	0	0	0	1	1	1

Рис. 4. Управляемый вероятностный тест $CRT(4, 6, 9)$
Fig. 4. Controlled random test $CRT(4, 6, 9)$

Приведенные выше правила позволяют строить управляемые вероятностные тесты $CRT(q, h, n)$ с заданными свойствами в рамках ограничений, определяемых взаимосвязью трех параметров, а именно количеством тестовых наборов q, их разрядностью n, а также минимальным значением расстояния Хэмминга h между любыми двумя наборами теста. Например, для получения теста $CRT(4, 6, 9)$ нет необходимости перебора потенциальных $n = 9$ разрядных кандидатов в тесты и выбора таких из них, для которых $h \geq 6$. Эту задачу можно свести к задаче построения управляемого вероятностного теста $CRT(4, 6, 9)$ на базе шаблонов (см., например, рис. 4) и получения семейства таких тестов, используя правила, приведенные выше.

Для обеспечения большей гибкости при построении управляемых вероятностных тестов, в особенности для заданных величин n, приведем следующее правило, также вытекающее из фундаментальных основ теории помехоустойчивого кодирования [22, 23]. Данное правило можно назвать правилом исключения, которое формулируется следующим образом.

Правило 7. Результатом удаления одноименных бит (разрядов) во всех q тестовых наборах $T_0, T_1, \ldots, T_{q-1}$ теста $CRT(q, h, n)$ является тест $CRT(q, h–1, n–1)$.

Приведенное правило позволяет формировать требуемую разрядность тестовых наборов путем ее уменьшения в исходном тесте при одновременном уменьшении значения h. Например, удаление любого разряда наборов теста $CRT(4, 6, 9)$, приведенного на рис. 4, позволяет получить тест $CRT(4, 5, 8)$.

Очевидно, что наличие большого множества разнообразных шаблонов позволит эффективно решить задачу построения управляемых вероятностных тестов на основании приведенных правил с требуемым сочетанием их параметров q, h и n.

Процедура построения управляемых вероятностных тестов с фиксированным расстоянием Хэмминга. Как было отмечено в предыдущих разделах, существуют подходы к построению управляемых вероятностных тестов с малым числом тестовых наборов на основе формальных процедур применения правил синтеза без каких-либо значительных вычислительных затрат. Главной особенностью таких тестов является взаимосвязь между значением $h = \min HD(T_i, T_j)$ расстояния Хэмминга и количеством q тестовых наборов. Как было показано ранее, увеличение требуемого значения $h = \min HD(T_i, T_j)$ расстояния Хэмминга при генерировании теста $CRT(q, h, n)$ уменьшает число q его наборов.

В качестве альтернативы известным решениям предлагается подход, основанный на увеличении числа тестовых наборов q в тесте при сохранении величины $\min HD(T_i, T_j)$ на приемлемом уровне. Основным отличием предлагаемого подхода является использование предложенной авторами меры различия $MD(T_i, T_j)$, основанной на определении расстояния Хэмминга (1) для тестовых наборов, состоящих из символов различных алфавитов. Эта мера вычисляется для двух двоичных наборов $T_i = t_{i,0}\, t_{i,1}\, t_{i,2} \ldots t_{i,n-1}$ и $T_j = t_{j,0}\, t_{j,1}\, t_{j,2} \ldots t_{j,n-1}$, где $t_{i,l}, t_{j,l} \in \{0, 1\}$, $l \in \{0, 1, 2, \ldots, n-1\}$, и состоит из n компонент $HD_1, HD_2, HD_3, \ldots, HD_n$, которые показывают различие этих наборов. Свойство 4 данной меры констатирует, что ее компоненты связаны соотношением $HD_1 \geq HD_2 \geq HD_3 \geq \ldots, \geq HD_n$, где $HD_r = HD_r[T_i(r), T_j(r)]$ для $r \in \{1, 2, 3, \ldots, n\}$. Согласно определению 2 наборы $T_i(r)$ и $T_j(r)$ являются представлением двоичных наборов T_i и T_j символами 2^r-ичной системы счисления, включающей 2^r символов.

На основании свойства 4 меры различия $MD(T_i, T_j)$ сформулируем следующее утверждение, которое составляет основу предлагаемой процедуры построения управляемых вероятностных тестов с небольшим количеством q тестовых наборов и заданным значением $\min HD(T_i, T_j)$.

Утверждение 2. *Управляемый вероятностный тест, состоящий из* $q = 2^r$ *двоичных наборов, где* r *является минимальным значением из множества* $\{1, 2, 3, \ldots, n\}$, *для которого* $HD_r[T_i(r), T_j(r)] = \max HD_r[T_i(r), T_j(r)]$ *для всех* $i \neq j \in \{0, 1, 3, \ldots, q-1\}$ *и* $n \bmod r = 0$, *имеет значение расстояния Хэмминга* $h = \min HD(T_i, T_j) = n/r$.

Ограниченное количество $q = 2^r$ тестовых наборов определяется ограниченным количеством 2^r символов алфавита, в котором представлены тестовые наборы $T_i(r) = t_{i,0}(r)\, t_{i,1}(r)\, t_{i,2}(r) \ldots t_{i,n/r-1}(r)$ и $T_j(r) = t_{j,0}(r)\, t_{j,1}(r)\, t_{j,2}(r) \ldots t_{j,n/r-1}(r)$. Только в таком случае символы одних и тех же цифр (разрядов) во всех q тестовых наборах могут принимать разные значения без повторений. Это является условием достижения максимального значения $HD_r[T_i(r), T_j(r)]$ расстояния Хэмминга для всех пар тестовых наборов $T_i(r)$ и $T_j(r)$ при $i \neq j \in \{0, 1, 2, \ldots, q-1\}$. Поясним суть данного утверждения на примере управляемого вероятностного теста (табл. 4). Исходный тест состоит из $q = 4$ шестиразрядных тестовых наборов $T_i(1) = t_{i,0}(1)\, t_{i,1}(1)\, t_{i,2}(1)\, t_{i,3}(1)\, t_{i,4}(1)\, t_{i,5}(1)$, которые также представлены в четверичной $T_i(2) = t_{i,0}(2)\, t_{i,1}(2)\, t_{i,2}(2)$ и восьмеричной $T_i(3) = t_{i,0}(3)\, t_{i,1}(3)$ системах счисления.

Таблица 4
Двоичный управляемый вероятностный тест для $n = 6$ и его интерпретации

Table 4
Binary controlled random test for n = 6 and its interpretations

$T_i(r)$	$T_i(1) = t_{i,0}(1)t_{i,1}(1)t_{i,2}(1)t_{i,3}(1)t_{i,4}(1)t_{i,5}(1)$	$T_i(2) = t_{i,0}(2)\, t_{i,1}(2)\, t_{i,2}(2)$	$T_i(3) = t_{i,0}(3)\, t_{i,1}(3)$
T_0	1 1 1 0 0 1	3 2 1	7 1
T_1	0 1 0 0 0 0	1 0 0	2 0
T_2	1 0 0 1 1 1	2 1 3	4 7
T_3	0 0 1 1 1 0	0 3 2	1 6

Как видно из табл. 4, во всех разрядах четверичного и восьмеричного представления тестовых наборов нет повторяющихся символов. Это говорит о том, что в обоих случаях расстояние Хэмминга между тестовыми наборами согласно (1) принимает максимальные значения. Действительно, для любых двух наборов T_i и T_j теста $HD_2[T_i(2), T_j(2)] = \max HD_2[T_i(2), T_j(2)] = n/2 = 3$,

а также $HD_3[T_i(3), T_j(3)] = \max HD_3[T_i(3), T_j(3)] = n/3 = 2$, причем для четверичного случая в каждом разряде $t_{i,0}(2)\ t_{i,1}(2)\ t_{i,2}(2)$ тестовых наборов T_0, T_1, T_2, T_3 используются все четыре символа 0, 1, 2 и 3 без повторений. В результате для исходного двоичного теста имеем $HD_1[T_0(1), T_1(1)] = 3$, $HD_1[T_0(1), T_2(1)] = 4$, $HD_1[T_0(1), T_3(1)] = 5$, $HD_1[T_1(1), T_2(1)] = 5$, $HD_1[T_1(1), T_3(1)] = 4$, $HD_1[T_2(1), T_3(1)] = 3$, что соответствует утверждению 2.

Аналогичный пример приведен в табл. 5 для $n = 12$, в котором для восьмеричного представления тестовых наборов $T_i(3)$ в каждом из четырех их разрядов используются все восемь символов 0, 1, 2, 3, 4, 5, 6 и 7 без повторений.

Таблица 5
Двоичный управляемый вероятностный тест для $n = 12$ и его интерпретации

Table 5
Binary controlled random test for n = 12 and its interpretations

$T_i(r)$	$T_i(1)$	$T_i(2)$	$T_i(3)$	$T_i(4)$
T_0	1 1 0 1 0 1 0 0 1 0 0 0	3 1 1 0 2 0	6 5 1 0	D 4 8
T_1	1 0 0 0 1 1 0 1 0 1 1 1	2 0 3 1 1 3	4 3 2 7	8 D 7
T_2	0 0 0 1 1 1 0 1 1 1 1 0	0 1 3 1 3 2	0 7 3 6	1 D E
T_3	0 0 1 0 0 0 0 0 0 0 0 1	0 2 0 0 0 1	1 0 0 1	2 0 1
T_4	0 1 0 1 1 0 1 0 0 0 1 0	1 1 2 2 0 2	2 6 4 2	5 A 2
T_5	1 1 1 0 0 1 1 1 1 0 1 1	3 2 1 3 2 3	7 1 7 3	E 7 B
T_6	0 1 1 0 1 0 1 0 1 1 0 0	1 2 2 2 3 0	3 2 5 4	6 A C
T_7	1 0 1 1 0 0 1 1 0 1 0 1	2 3 0 3 1 1	5 4 6 5	B 3 5

Руководствуясь утверждением 2, заключаем, что управляемый вероятностный тест, состоящий из $2^r = 2^3$ двоичных тестовых наборов, включающих по $n = 12$ бит, имеет $h = \min HD(T_i, T_j) = n/r = 12/3 = 4$. Соответственно, тест, представленный в табл. 5, можно описать как $CRT(q, h, n) = CRT(8, 4, 12)$, а в табл. 4 – как $CRT(4, 3, 6)$.

Основой построения тестов, приведенных в указанных таблицах, являются шаблоны $CRT(2^r, 1, r)$, которые представляют собой полный набор 2^r символов в одной из систем счисления, определяемой количеством бит r, используемых для их кодирования. Для синтеза теста, представленного в табл. 4, применялся шаблон $CRT(2^2, 1, 2)$, а для второго теста – $CRT(2^3, 1, 3)$. Первый шаблон $CRT(2^2, 1, 2)$ состоит из четырех ($q = 2^2$) четверичных цифр 0, 1, 2 и 3, сгенерированных в произвольном порядке, которые представляют собой управляемый вероятностный тест, состоящий из четырех двухразрядных ($n = 2$) двоичных наборов. Значение $h = \min HD(T_i, T_j)$ для этого шаблона равняется единице. Аналогична структура и шаблона $CRT(2^3, 1, 3)$, кроме количества тестовых наборов, равного восьми, и их разрядности, равной трем. С применением правил 1–3 на основании исходных шаблонов можно построить их семейства с теми же характеристиками q, h и n.

На основании применения шаблонов $CRT(2^r, 1, r)$ и ранее рассмотренных правил преобразования шаблонов и тестов можно построить управляемые вероятностные тесты с заданными характеристиками q, h и n. Использование правил повторения и объединения (правил 4 и 5) для исходных шаблонов $CRT(2^r, 1, r)$ позволяет строить тесты $CRT(2^r, w, w \cdot r)$.

Ограничение ($n \bmod r = 0$), накладываемое на разрядность тестовых наборов и заключающееся в обеспечении делимости n на r, определяет зависимость между $h = \min HD(T_i, T_j)$ и n теста $CRT(2^r, w, w \cdot r)$. Расширение области применения шаблонов $CRT(2^r, 1, r)$ для произвольной разрядности n позволяет строить тесты $CRT(2^r, \lfloor n/r \rfloor, n)$.

На основании утверждения 2 можно предложить формальную процедуру построения управляемых вероятностных тестов с $q = 2^r$ двоичными наборами и заданным значением $\min HD(T_i, T_j) \geq \lfloor n/r \rfloor$. Возможные фиксированные значения величины $\min HD(T_i, T_j)$ зависят от числа n бит двоичных тестовых наборов T_i и T_j. Например, для $n = 32$ возможные варианты тестов с заданным значением $h = \min HD(T_i, T_j) \geq \lfloor n/r \rfloor$ и числом q тестовых наборов представлены в табл. 6.

Таблица 6
Зависимость между числом бит $n = 32$ двоичных наборов и $h = \min HD(T_i, T_j)$
Table 6
Dependence between the number of bits n = 32 of binary patterns and $h = \min HD(T_i, T_j)$

r	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	…	32
$h = \lfloor 32/r \rfloor$	16	10	8	6	5	4	4	3	3	2	2	2	2	2	2	1	1	…	1
q	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}	2^{18}	…	2^{32}

Как видно из табл. 6, фиксированное значение n разрядности тестовых наборов предопределяет значения $\min HD(T_i, T_j)$, для которых возможно построение теста на основе утверждения 2. Очевидно, что наибольший интерес представляют случаи, для которых $\min HD(T_i, T_j)$ принимает приемлемо большие значения, т. е. для малых значений r и относительно небольшого количества $q = 2^r$ тестовых наборов.

Процедура построения управляемых вероятностных тестов с фиксированным расстоянием Хэмминга, исходными данными для которой являются разрядность n двоичных тестовых наборов и требуемое значение $\min HD(T_i, T_j)$, включает следующие шаги:

1. Определяется максимальное значение $r \in \{1, 2, 3, \ldots, n\}$, для которого выполняется неравенство $\min HD(T_i, T_j) \le \lfloor n/r \rfloor$. Это значение согласно утверждению 2 определяет число q наборов в тесте, равное 2^r, и расстояние Хэмминга $HD(T_i, T_j) \ge \lfloor n/r \rfloor$ для всех $i \ne j \in \{0, 1, 3, \ldots, 2^r - 1\}$.

2. Значения первых r бит 2^r тестовых наборов $T_0, T_1, T_2, \ldots, T_{q-1}$ устанавливаются равными двоичному коду одного из символов алфавита, содержащего 2^r символов. Двоичные коды символов формируются случайным образом без повторений, начиная с первого набора T_0 до последнего T_{q-1}. Для этих целей можно использовать шаблоны из семейства шаблонов $CRT(2^r, 1, r)$. Таким образом, каждый тестовый набор будет содержать в первых r разрядах уникальную двоичную комбинацию, соответствующую одному из 2^r символов.

3. Шаг 2 повторяется $\lfloor n/r \rfloor - 1$ раз для всех последующих r разрядных блоков двоичных тестовых наборов, т. е. на второй итерации задаются уникальные r-битовые коды следующих r разрядов, а именно $r, r+1, \ldots, 2r-1$ бит тестовых наборов.

4. Значения оставшихся $n - \lfloor n/r \rfloor \cdot r$ разрядов, если таковые имеются, всех тестовых наборов формируются случайным образом.

Расширением данной процедуры может быть выбор не обязательно последовательных r разрядов формируемых тестовых наборов, а любых произвольных r из n разрядов для формирования в них двоичных кодов символов, что эквивалентно применению правила 2 к результату, полученному согласно приведенной процедуре. Единственным ограничением является требование выбора непересекающихся блоков по r бит.

Приведем пример применения рассмотренной процедуры для синтеза управляемого вероятностного теста для $n = 16$ и $\min HD(T_i, T_j) = 5$:

1. На основании неравенства $\min HD(T_i, T_j) = 5 \le \lfloor n/r \rfloor = \lfloor 16/r \rfloor$ получаем значение $r = 3$, так как оно является максимальным значением r из множества его значений $\{1, 2, 3, \ldots, 16\}$, при котором выполняется данное неравенство. Соответственно, формируемый тест будет состоять из $2^r = 2^3 = 8$ наборов.

2. Значения первых трех разрядов $t_{i,0}$, $t_{i,1}$ и $t_{i,2}$ тестовых наборов $T_0, T_1, T_2, \ldots, T_7$ устанавливаются равными одному из двоичных кодов 000, 001, 010, …, 111 символов 0, 1, 2, …, 7 восьмеричного алфавита. Для этого можно использовать один из шаблонов $CRT(2^3, 1, 3)$, например 010, 101, 011, 000, 111, 001, 110 и 100. Таким образом, каждый тестовый набор будет содержать в первых трех разрядах уникальную двоичную комбинацию (табл. 7).

3. Шаг 2 повторяется $\lfloor 16/3 \rfloor - 1 = 4$ раза. Для каждого блока, состоящего из трех бит, значения символов восьмеричного алфавита назначаются случайным образом без повторений.

4. Значения оставшегося $16 - \lfloor 16/3 \rfloor \cdot 3 = 1$ бита $t_{i,15}$ всех тестовых наборов формируются случайным образом.

В результате применения рассмотренной процедуры получен тест $CRT(8, 5, 16)$, представленный в табл. 7, для которого выполняется следующее условие: $HD(T_i, T_j) \ge \min HD(T_i, T_j) = 5$. Из табл. 7 следует, что все значения $HD(T_i, T_j)$ больше или равны пяти.

Таблица 7
Управляемый вероятностный тест *CRT*(8, 5, 16) с min$HD(T_i, T_j) = 5$ и $n = 16$

Table 7
Controlled random test CRT(8, 5, 16) *with* min$HD(T_i, T_j) = 5$ *and* $n = 16$

T	$t_{i,0}$	$t_{i,1}$	$t_{i,2}$	$t_{i,3}$	$t_{i,4}$	$t_{i,5}$	$t_{i,6}$	$t_{i,7}$	$t_{i,8}$	$t_{i,9}$	$t_{i,10}$	$t_{i,11}$	$t_{i,12}$	$t_{i,13}$	$t_{i,14}$	$t_{i,15}$
T_0	0	1	0	0	0	0	0	1	1	1	1	1	0	1	0	0
T_1	1	0	1	1	1	0	0	1	0	0	1	1	1	1	1	1
T_2	0	1	1	1	0	0	1	0	1	1	0	0	1	0	0	1
T_3	0	0	0	1	1	1	0	0	0	0	1	0	0	0	1	1
T_4	1	1	1	0	0	1	1	1	0	1	0	1	1	1	0	0
T_5	0	0	1	0	1	0	0	0	1	0	0	1	1	0	1	1
T_6	1	1	0	1	0	1	1	0	0	0	0	0	0	1	1	0
T_7	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0

Результаты экспериментальных исследований. Для подтверждения полученных результатов был проведен ряд вычислительных и практических экспериментов. В качестве объекта исследования рассматривалось запоминающее устройство (ЗУ) и его наиболее сложные для обнаружения кодочувствительные неисправности [6, 21, 31, 32]. Анализировались пассивные кодочувствительные неисправности *PNPSFk*, где *k* обозначает количество произвольных ячеек ЗУ емкостью *n* бит, участвующих в конкретной неисправности, одна из которых является базовой. Отметим, что результаты, полученные для *PNPSFk*, легко обобщаются и на другие классы кодочувствительных и иных неисправностей ЗУ в силу того, что *PNPSFk* является наиболее трудно обнаруживаемой разновидностью неисправностей ЗУ [6].

В качестве теста ЗУ использовался широко применяемый на практике маршевый тест *MATS++* [6, 31, 32]. Его неразрушающая версия $\{\uparrow(Rt, W\bar{t}); \downarrow(R\bar{t}, Wt, Rt)\}$ состоит из двух последовательных фаз, каждая из которых содержит операции чтения (*R*) из ячейки ЗУ и записи (*W*) в ячейку ее содержимого $t \in \{0,1\}$ или инверсного значения $\bar{t}$. Символ ↑ обозначает возрастающую последовательность адресов ячеек ЗУ, а символ ↓ – убывающую [6]. В работе [31] показано, что, основываясь на наличии степеней свободы маршевых тестов, в качестве начального состояния ЗУ может быть использовано любое начальное состояние его ячеек. Независимо от первоначального состояния ЗУ полнота покрытия маршевого теста $FC_{March\_Test}(PNPSFk)$, определяемая как отношение обнаруженных неисправностей к их общему числу, принимает постоянное значение. Например, для *PNPSF*3 и *PNPSF*5 эти значения равняются *FC*(*PNPSF*3) = 25 % и *FC*(*PNPSF*5) = 6,25 % [32]. Эффективность многократного применения маршевого теста, в том числе и теста *MATS++*, при случайных начальных состояниях ЗУ оценивается соотношением

$$FC_{MATS++}(PNPSFk, q) = \left(1 - \left(1 - \frac{FC_{MATS++}(PNPSFk)}{100\ \%}\right)^q\right) \cdot 100\ \%. \qquad (3)$$

Первоначально рассматривался случай ЗУ, состоящего из $n = 12$ ячеек, и многократный (*q*-кратный) маршевый тест *MATS++*, используемый для обнаружения его *PNPSF*3 неисправностей. В качестве начальных состояний ЗУ применялись как управляемые вероятностные тесты (*CRT*), синтезированные согласно процедуре, предложенной авторами, так и равномерно распределенные случайные значения (*Random*) содержимого ячеек ЗУ. В первом столбце табл. 8 приведены управляемые вероятностные тесты *CRT*(4, 6, 12), *CRT*(8, 4, 12) и *CRT*(16, 3, 12), используемые в качестве начальных состояний ЗУ при обнаружении неисправностей *PNPSF*3 маршевым тестом *MATS++* в ЗУ с $n = 12$ ячейками.

Обозначение *CRT*(8, 4, 12), приведенное в табл. 8, описывает управляемый вероятностный тест с $h = 4$, состоящий из $q = 8$ наборов, представленных 12 двоичными разрядами. Например, $T_1 = 6401_{(8)}$, заданный в восьмеричном алфавите, соответствует его двоичному представлению $11010000\ 0001_{(2)}$ (табл. 8). Во втором столбце представлены сгенерированные случайным образом по равномерному закону распределения начальные состояния ЗУ, соответствующие веро-

ятностному тесту. В последующих столбцах приведены значения полноты покрытия для случая управляемого вероятностного теста (FC_{CRT}), вероятностного теста (FC_{Random}) и FC_{MATS++}, определенного согласно соотношению (3).

Таблица 8
Сравнительная оценка тестов $CRT(q, h, 12)$ для $h = \min HD(T_i, T_j) = 6$, 4 и 3

Table 8
Comparative evaluation of tests $CRT(q, h, 12)$ *for* $h = \min HD(T_i, T_j) = 6$, 4 and 3

$CRT(4, 6, 12) = T_0, T_1, T_2, T_4$	*Random*	FC_{CRT}	FC_{Random}	FC_{MATS++}
232002, 310231, 021113, 103320	231233, 033302, 200330, 001210	85,61 %	68,94 %	68,36 %
$CRT(8, 4, 12) = T_0, T_1, T_2, \ldots T_8$	*Random*	FC_{CRT}	FC_{Random}	FC_{MATS++}
3640, 6401, 0276, 5065, 4523, 1712, 7357, 2134	2771, 1721, 5725, 2466, 3140, 0142, 1700, 6134	98,48 %	89,39 %	89,99 %
$CRT(16, 3, 12) = T_0, T_1, T_2, \ldots T_{16}$	*Random*	FC_{CRT}	FC_{Random}	FC_{MATS++}
06C, D0F, EFD, 6E8, 8A7, 7B6, 91A, 2C0, 58B, F92, C45, 124, 4DE, B71, A33, 359	114, 9E3, 69B, E0B, BAE, 2A3, F9D, CBF, 4E7, 7B1, FCB, A8C, 545, 0E2, 338, 1D9	100 %	99,62 %	98,99 %

Аналогичные результаты приведены в табл. 9 для случая ЗУ с 24 ячейками.

Таблица 9
Сравнительная оценка тестов $CRT(q, h, 24)$ для $h = \min HD(T_i, T_j) = 12$, 8 и 6

Table 9
Comparative evaluation of tests $CRT(q, h, 24)$ *for* $h = \min HD(T_i, T_j) = 12$, 8 and 6

$CRT(4, 12, 24) = T_0, T_1, T_2, T_4$	*Random*	FC_{CRT}	FC_{Random}	FC_{MATS++}
203331131202, 321012222311, 130123010030, 012200303123	010101132120, 033011003010, 023033231202, 010113213031	84,45 %	66,01 %	68,36 %
$CRT(8, 8, 24) = T_0, T_1, T_2, \ldots T_8$	*Random*	FC_{CRT}	FC_{Random}	FC_{MATS++}
55637371, 60706167, 27320502, 01275736, 33144640, 46463214, 72512455, 14051023	56074427, 44157507, 73750741, 67135517, 24470321, 21770371, 47707356, 67464234	98,67 %	88,94 %	89,99 %
$CRT(16, 6, 24) = T_0, T_1, T_2, \ldots T_{16}$	*Random*	FC_{CRT}	FC_{Random}	FC_{MATS++}
E36BB3, BFF53E, F6DFC8, 8A9649, 9B739A, 650D25, 07C18C, 2C545F, D0ECD7, A2A872, 391A1D, 443E60, C120FB, 1D8901, 5EB7A6, 7842E4	031121, B4F64D, E3E3B1, 44DA93, B5B6E5, A3FB8E, D80ABC, 2C2CCF, 098D2B, B9E2AD, 23B3E5, F435C3, 458A8E, 912087, 11ED17, 3A3B69	100 %	98,43 %	98,99 %

Усредненные значения полноты покрытия FC_{CRT} и FC_{Random} для *PNPSF*3 по 100 результатам, аналогичным данным табл. 8 и 9, представлены на рис. 5.

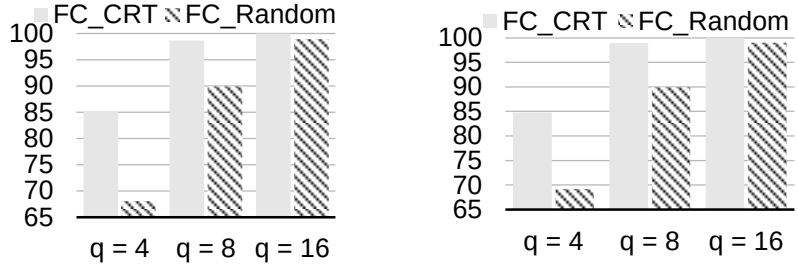


Рис. 5. Усредненные значения полноты покрытия: *a)* $n = 12$; *b)* $n = 24$
Fig. 5. Average values of fault coverage: a) $n = 12$; *b)* $n = 24$

Приведенные в табл. 8, 9 и на рис. 5 результаты подтверждают гипотезу о существенном преимуществе управляемых вероятностных тестов над вероятностными тестами, заключающемся в большей эффективности обнаружения сложных неисправностей ЗУ, таких как *PNPSFk*. Применение такого же теста *CRT*(16, 6, 12) для обнаружения неисправностей *PNPSF*5 в ЗУ с 12 ячейками также показало преимущество управляемых вероятностных тестов. В этом случае FC_{CRT} = 24,85 %, а FC_{Random} = 22,84 % при теоретическом значении FC_{MATS+} = 22,75 %.

Заметно большая покрывающая способность наблюдается для случая, когда q принимает небольшие значения, соответствующие r = 2 и 3. Величина q не зависит от размерности n ЗУ.

Заключение. Рассмотрены меры различия, основанные на применении модификаций определения расстояния Хэмминга и отображении двоичных тестовых наборов в виде последовательностей символов, представленных в различных алфавитах. Новые меры различия расширяют возможности генерирования тестовых последовательностей при формировании управляемых вероятностных тестов. В качестве альтернативы известным решениям предлагается подход, основанный на увеличении числа тестовых наборов в тесте при сохранении величины минимального значения расстояния Хэмминга между наборами на приемлемом уровне. Показано, что достижение максимального значения расстояния Хэмминга для наборов, представленных большим количеством двоичных символов, обеспечивает такое же значение расстояния для случая, когда символы задаются меньшим числом бит. Это позволяет строить управляемые вероятностные тесты без необходимости перечисления кандидатов в тестовые наборы, что сводит задачу синтеза управляемого вероятностного теста к формальной процедуре, предложенной в настоящей статье.

Вклад авторов. *В. Н. Ярмолик* предложил процедуру построения управляемых вероятностных тестов, основанную на применении модифицированных методов вычисления расстояния Хэмминга. *В. В. Петровская* провела экспериментальные исследования. *В. В. Петровская*, *Д. В. Деменковец* и *В. А. Леванцевич* приняли участие в обобщении, анализе и оформлении полученных результатов.

Список использованных источников

1. Orso, A. Software testing: A research travelogue (2000–2014) / A. Orso, G. Rothermel // Proc. of Future of Software Engineering Proceeding (FOSE'14), Hyderabad, India, 31 May – 7 June 2014. – Hyderabad, 2014. – P. 117–132.

2. An orchestrated survey on automated software test case generation / S. Anand, E. Burke, T. Chen [et al.] // Journal of Systems and Software. – 2014. – Vol. C-39, no. 4. – P. 582–586.

3. Arcuri, A. Random testing: Theoretical results and practical implications / A. Arcuri, M. Z. Iqbal, L. Briand // IEEE Transactions on Software Engineering. – 2011. – Vol. 38, no. 2. – P. 258–277.

4. Malaiya, Y. K. The coverage problem for random testing / Y. K. Malaiya, S. Yang // Proc. of the Intern. Test Conf., Philadelphia, PA, USA, 16–18 Oct. 1984. – Philadelphia, 1984. – P. 237–242.

5. Duran, J. W. An evaluation of random testing / J. W. Duran, S. C. Ntafos // IEEE Transactions on Software Engineering. – 1984. – Vol. SE-10, no. 4. – P. 438–444.

6. Ярмолик, В. Н. Контроль и диагностика вычислительных систем / В. Н. Ярмолик. – Минск : Бестпринт, 2019. – 387 с.

7. Malaiya, Y. K. Antirandom testing: getting the most out of black-box testing / Y. K. Malaiya // Proc. of the Intern. Symp. on Software Reliability Engineering, Toulose, France, 24–27 Oct. 1995. – Toulose, 1995. – P. 86–95.

8. A survey on adaptive random testing / R. Huang, W. Sun, Y. Xu [et al.] // IEEE Transactions on Software Engineering. – 2021. – Vol. 47, no. 10. – P. 2052–2083.

9. Adaptive random testing: The art of test case diversity / T. Y. Chen, F. C. Kuo, R. G. Merkel, T. H. Tse // Journal of Systems and Software. – 2010. – Vol. 83. – P. 60–66.

10. Antirandom testing: A distance-based approach / S. N. Wu, S. Jandhyala, Y. K. Malaiya, A. P. Jayasumana // Hindawi Publishing Corporation VLSI Design. – 2008. – Vol. 2008, article ID 165709. – 9 p. – DOI: 10.1155/2008/165709.

11. Fast antirandom (FAR) test generation / A. von Mayrhauser, A. Bai, T. Chen [et al.] // Proc. of the Third IEEE Intern. High-Assurance System Engineering Symp., Washington, D.C., USA, 13–14 Nov. 1998. – Washington, 1998. – P. 262–269.

12. Xu, S. Orderly random testing for both hardware and software / S. Xu // Proc. of the 2008 14th IEEE Pacific Rim Intern. Symp. on Dependable Computing, Washington, D.C., USA, 15–17 Dec. 2008. – Washington, 2008. – P. 160–167.

13. Xu, S. Maximum distance testing / S. Xu, J. Chen // Proc. of the 11th IEEE Asian Test Symp. (ATS'02), Guam, USA, 18–20 Nov. 2002. – Guam, 2002. – P. 15–20.

14. Kuo, F. C. An indepth study of mirror adaptive random testing / F. C. Kuo // Proc. of the Ninth Intern. Conf. on Quality Software (QSIC 2009), Jeju, Korea (South), 24–25 Aug. 2009. – Jeju, 2009. – P. 51–58.

15. Tappenden, A. A novel evolutionary approach for adaptive random testing / A. Tappenden, J. Miller // IEEE Transactions on Reliability. – 2009. – Vol. 58, no. 4. – P. 619–633.

16. Zhibo, Li. An enhanced adaptive random testing by dividing dimensions independently / Li Zhibo, Li Qingbao, Yu Lei // Mathematical Problems in Engineering. – 2019. – Vol. 2019. – P. 1–15.

17. Nikravan, E. Hybrid adaptive random testing / E. Nikravan, S. Parsa // International Journal of Computing Science & Mathematics. – 2020. – Vol. 11, no. 3. – P. 209.

18. Yarmolik, S. V. Controlled random tests / S. V. Yarmolik, V. N. Yarmolik // Automation and Remote Control. – 2012. – Vol. 73, no. 10. – P. 1704–1714.

19. Mrozek, I. Antirandom test vectors for BIST in hardware/software systems / I. Mrozek, V. N. Yarmolik // Fundamenta Informaticae. – 2012. – Vol. 119, no. 2. – P. 163–185.

20. Mrozek, I. Multiple controlled random testing / I. Mrozek, V. N. Yarmolik // Fundamenta Informaticae. – 2016. – Vol. 144, no. 1. – P. 23–43.

21. Ярмолик, С. В. Итеративные почти псевдоисчерпывающие вероятностные тесты / С. В. Ярмолик, В. Н. Ярмолик // Информатика. – 2010. – № 2(26). – С. 66–75.

22. Hamming, R. W. Error detecting and error correcting codes / R. W. Hamming // The Bell System Technical Journal. – 1950. – Vol. 29, no. 2. – P. 147–160.

23. Peterson, W. W. Error-Correction Codes / W. W. Peterson, E. J. Weldon. – Cambridge, Massachusetts, London, England : The MIT Press, 1972. – 560 p.

24. Садовский, М. Г. О сравнении символьных последовательностей / М. Г. Садовский // Вычислительные технологии. – 2005. – № 3(10). – С. 106–116.

25. Ярмолик, В. Н. Модификации способов определения расстояния Хэмминга для их применения в качестве мер различия при генерировании управляемых вероятностных тестов / В. Н. Ярмолик, В. В. Петровская, Н. А. Шевченко // Информатика. – 2024. − Т. 21, № 2. – С. 54–72.

26. Многомерный портрет цифровых последовательностей идеального «белого шума» в свертках Хэмминга / В. И. Волчихин, А. И. Иванов, А. П. Юнин, Е. А. Малыгина // Информатика, вычислительная техника и управление. – 2017. – № 4(1). – С. 4–12.

27. Ярмолик, В. Н. Мера различия для управляемых вероятностных тестов / В. Н. Ярмолик, В. В. Петровская, Н. А. Шевченко // Доклады БГУИР. – 2024. – № 4(22). – С. 76–83.

28. Plotkin, M. Binary codes with specified minimum distance / M. Plotkin // IRE Transactions on Information Theory. – 1960. – Vol. 6, no. 4. – P. 445–450.

29. Yarmolik, S. V. The synthesis of probability tests with a small number of kits / S. V. Yarmolik, V. N. Yarmolik // Automatic Control and Computer Sciences. – 2011. – Vol. 45, no. 3. – P. 133–141.

30. Mrozek, I. Optimal controlled random tests / I. Mrozek, V. Yarmolik // Proc. of Computer Information Systems and Industrial Management: 16th IFIP TC8 Intern. Conf., CISIM 2017, Bialystok, Poland, 16–18 June 2017. – Bialystok, 2017. – P. 27–38.

31. Sokol, B. Memory faults detection techniques with use of degrees of freedom in march tests / B. Sokol, V. N. Yarmolik // Proc. of IEEE East-West Design & Test Workshop (EWDTW'05), Odessa, Ukraine, 15–19 Sept. 2005. – Odessa, 2005. – P. 96–101.

32. Ярмолик, С. В. Обнаружение кодочувствительных неисправностей запоминающих устройств с многократным использованием маршевых тестов / С. В. Ярмолик, В. Н. Ярмолик // Информатика. – 2006. − Т. 19, № 1. – С. 104–113.

References

1. Orso A., Rothermel G. Software testing: A research travelogue (2000–2014). *Proceeding of Future of Software Engineering Proceeding (FOSE'14), Hyderabad, India, 31 May – 7 June 2014.* Hyderabad, 2014, pp. 117–132.

2. Anand S., Burke E. K., Chen T. Y., Clark J., Cohen M. B., …, Zhu H. An orchestrated survey on automate software test case generation. *Journal of Systems and Software*, 2014, vol. C-39, no. 4, pp. 582–586.

3. Arcuri A., Iqbal M. Z., Briand L. Random testing: Theoretical results and practical implications. *IEEE Transactions on Software Engineering*, 2011, vol. 38, no. 2, pp. 258–277.

4. Malaiya Y. K., Yang S. The coverage problem for random testing. *Proceeding of the International Test Conference, Philadelphia, PA, USA, 16–18 October 1984.* Philadelphia, 1984, pp. 237–242.

5. Duran J. W., Ntafos S. C. An evaluation of random testing. *IEEE Transactions on Software Engineering*, 1984, vol. SE-10, no. 4, pp. 438–444.

6. Yarmolik V. N. Control' i diagnostika vuchislitel'nuch system. *Computer Systems Testing and Diagnoses.* Minsk, Bestprint, 2019, 387 p. (In Russ.).

7. Malaiya Y. K. Antirandom testing: getting the most out of black-box testing. *Proceedings of the International Symposium on Software Reliability Engineering, Toulose, France, 24–27 October 1995*. Toulose, 1995, pp. 86–95.

8. Huang R., Sun W., Xu Y., Chen H., Towey D., Xia X. A survey on adaptive random testing. *IEEE Transactions on Software Engineering*, 2021, vol. 47, no. 10, pp. 2052–2083.

9. Chen T. Y., Kuo F. C., Merkel R. G., Tse T. H. Adaptive random testing: The art of test case diversity. *Journal of Systems and Software*, 2010, vol. 83, pp. 60–66.

10. Wu S. H., Jandhyala S., Malaiya Y. K., Jayasumana A. P. Antirandom testing: A distance-based approach. *Hindawi Publishing Corporation VLSI Design*, 2008, vol. 2008, article ID 165709, 9 p. DOI: 10.1155/2008/165709.

11. Von Mayrhauser A., Bai A., Chen T., Anderson C., Hajjar A. Fast antirandom (FAR) test generation. *Proceedings of the Third IEEE International High-Assurance System Engineering Symposium, Washington, D.C., USA, 13–14 November 1998.* Washington, 1998, pp. 262–269.

12. Xu S. Orderly random testing for both hardware and software. *Proceedings of the 2008 14th IEEE Pacific Rim International Symposium on Dependable Computing, Washington, D.C., USA, 15–17 December 2008*. Washington, 2008, pp. 160–167.

13. Xu S., Chen J. Maximum distance testing. *Proceedings of the 11th Asian Test Symposium (ATS'02), Guam, USA, 18–20 November 2002.* Guam, 2002, pp. 15–20.

14. Kuo F. C. An indepth study of mirror adaptive random testing. *Proceedings of the Ninth International Conference on Quality Software (QSIC 2009), Jeju, Korea (South), 24–25 August 2009*. Jeju, 2009, pp. 51–58.

15. Tappenden A., Miller J. A novel evolutionary approach for adaptive random testing. *IEEE Transactions on Reliability*, 2009, vol. 58, no. 4, pp. 619–633.

16. Zhibo Li, Qingbao Li, Lei Yu. An enhanced adaptive random testing by dividing dimensions independently. *Mathematical Problems in Engineering*, 2019, vol. 2019, pp. 1–15.

17. Nikravan E., Parsa S. Hybrid adaptive random testing. *International Journal of Computing Science & Mathematics*, 2020, vol. 11, no. 3, p. 209.

18. Yarmolik S. V., Yarmolik V. N. Controlled random tests. *Automation and Remote Control*, 2012, vol. 73, no. 10, pp. 1704–1714.

19. Mrozek I., Yarmolik V. Antirandom test vectors for BIST in hardware/software systems. *Fundamenta Informaticae*, 2012, vol. 119, no. 2, pp. 163–185.

20. Mrozek I., Yarmolik V. Multiple controlled random testing. *Fundamenta Informaticae*, 2016, vol. 144, no. 1, pp. 23–43.

21. Yarmolik S. V., Yarmolik V. N. *Iterative almost pseudo-exhaustive random tests*. Informatika [*Informatics*], 2010, vol. 26, no. 2, pp. 66−75 (In Russ.).

22. Hamming R. W. Error detecting and error correcting codes. *The Bell System Technical Journal*, 1950, vol. 29, no. 2, pp. 147–160.

23. Peterson W. W., Weldon E. J. *Error-Correction Codes*. Cambridge, Massachusetts, London, England, The MIT Press, 1972, 560 p.

24. Sadovskii M. G. *About symbolical sequences comparigion*. Vuchislitel'nue technologii [*Computational Technologise*], 2005, no. 3(10), pp. 106–116 (In Russ.).

25. Yarmolik V. N., Petrovskaya V. V., Shevchenko N. A. *Dissimilarity measures based on the application of Hamming distance to generate controlled probabilistic tests*. Informatika [*Informatics*], 2024, vol. 21, no. 2, pp. 54−72 (In Russ.).

26. Volchikhin V. I., Ivanov A. I., Yunin A. P., Malygina E. A. *A multidimensional picture of numerical sequences of the ideal "white noise" in Hamming convolutions*. Informatika, vychislitel'naya technika i upravlenie [*Computer Science, Computer Engineering and Control*], 2017, vol. 1, no. 4, pp. 4–12 (In Russ.).

27. Yarmolik V. N., Petrovskaya V. V., Shauchenka M. A. *Distance measure for controlled random tests*. Doklady BGUIR [*BSUIR Proceedings*], 2024, vol. 22, no. 4, pp. 76–83 (In Russ.).

28. Plotkin M. Binary codes with specified minimum distance. *IRE Transactions on Information Theory*, 1960, vol. 6, no. 4, pp. 445–450.

29. Yarmolik S. V., Yarmolik V. N. The synthesis of probability tests with a small number of kits. *Automatic Control and Computer Sciences*, 2011, vol. 45, no. 3, pp. 133–141.

30. Mrozek I., Yarmolik V. Optimal controlled random tests. *Proceedings of Computer Information Systems and Industrial Management: 16th IFIP TC8 International Conference, CISIM 2017, Bialystok, Poland, 16–18 June 2017.* Bialystok, 2017, pp. 27–38.

31. Sokol B., Yarmolik V. N. Memory faults detection techniques with use of degrees of freedom in march tests. *Proceedings of IEEE East-West Design & Test Workshop, Odessa, Ukraine, 15–19 September 2005.* Odessa, 2005, pp. 96–101.

32. Yarmolik S. V., Yarmolik V. N. *Detection of code-sensitive faults in memory devices using multiple march tests*. Informatika [*Informatics*], 2006, vol. 19, no. 1, pp. 104−113 (In Russ.).

Информация об авторах

Ярмолик Вячеслав Николаевич, доктор технических наук, профессор, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: yarmolik10ru@yahoo.com

Петровская Вита Владленовна, магистр технических наук, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: vita.petrovskaya@gmail.com

Деменковец Денис Викторович, магистр технических наук, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: demenkovets@bsuir.by

Леванцевич Владимир Александрович, магистр технических наук, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: lvn@bsuir.by

Information about the authors

Vyacheslav N. Yarmolik, D. Sc. (Eng.), Prof., Belarusian State University of Informatics and Radioelectronics.
E-mail: yarmolik10ru@yahoo.com

Vita V. Petrovskaya, M. Sc. (Eng.), Belarusian State University of Informatics and Radioelectronics.
E-mail: vita.petrovskaya@gmail.com

Denis V. Demenkovets, M. Sc. (Eng.), Belarusian State University of Informatics and Radioelectronics.
E-mail: demenkovets@bsuir.by

Vladimer A. Levantsevich, M. Sc. (Eng.), Belarusian State University of Informatics and Radioelectronics.
E-mail: lvn@bsuir.by

УДК 519.714.5
DOI: 10.37661/1816-0301-2025-22-1-27-39

Оригинальная статья
Original Article

Перепроектирование КМОП СБИС средствами инструмента синтеза Yosys

Д. И. Черемисинов, Л. Д. Черемисинова✉

Объединенный институт проблем информатики
Национальной академии наук Беларуси,
ул. Сурганова, 6, Минск, 220012, Беларусь
✉*E-mail: cld@newman.bas-net.by*

Аннотация

Цели. Рассматривается задача перепроектирования схемы транзисторного уровня, заданной в формате SPICE, в другом технологическом базисе. Целью статьи является разработка подхода к перепроектированию схем на основе использования средств программных пакетов автоматизации проектирования с открытым исходным кодом.

Методы. Предлагается метод, в основе которого лежат экстракция структуры на уровне логических элементов из плоского SPICE-описания транзисторной схемы и экспорт полученного иерархического SPICE-описания в программную среду открытого пакета синтеза Yosys. Целью экспорта являются преобразование описания логической сети в формате SPICE в описания на входных языках систем автоматизации проектирования, а также выполнение операций оптимизации и синтеза в среде Yosys.

Результаты. Для экспорта в ядро пакета Yosys логической сети, заданной в формате SPICE, была разработана программа на языке C++ с использованием классов пакета Yosys. Программа принимает и обрабатывает иерархическое SPICE-описание логической сети, переводя его в представление во внутреннем формате инструмента Yosys.

Заключение. Разработанная программа оформлена в виде программного модуля Yosys и интегрирована в его среду в качестве одной из команд. Над полученной модулем структурой логической сети могут быть выполнены все доступные в Yosys преобразования.

Ключевые слова: перепроектирование, декомпиляция транзисторных схем, КМОП-схемы, формат SPICE, язык Verilog, пакет Yosys

Для цитирования. Черемисинов, Д. И. Перепроектирование КМОП СБИС средствами инструмента синтеза Yosys / Д. И. Черемисинов, Л. Д. Черемисинова // Информатика. – 2025. − Т. 22, № 1. – С. 27–39. – DOI: 10.37661/1816-0301-2025-22-1-27-39.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 04.02.2025
Подписана в печать | Accepted 18.02.2025
Опубликована | Published 31.03.2025

Redesigning CMOS VLSI using Yosys synthesis tool

Dmitry I. Cheremisinov, Ljudmila D. Cheremisinova✉

The United Institute of Informatics Problems
of the National Academy of Sciences of Belarus,
st. Surganova, 6, Minsk, 220012, Belarus
✉*E-mail: cld@newman.bas-net.by*

Abstract
Objectives. The problem of reverse engineering of a transistor level circuit specified in the SPICE format in a different technological basis is considered. The goal of the work is to develop an approach to redesigning circuits using open source design automation software packages.
Methods. A method is proposed based on extracting the structure at the level of logical elements from a flat SPICE description of a transistor circuit and exporting the resulting hierarchical SPICE description to the software environment of the open synthesis package Yosys. The purpose of the export is to transform the description of the logical network in the SPICE format into descriptions in the input languages of design automation systems, as well as to perform optimization and synthesis operations in the Yosys environment.
Results. To export a logical network specified in the SPICE format to the core of the Yosys package, a program in C++ was developed using the classes of the Yosys package. The program accepts and processes the hierarchical SPICE description of the logical network, translating it into a representation in the internal format of the Yosys tool.
Conclusion. The developed program is designed as a Yosys program module and integrated into its environment as one of its commands. All the transformations available in Yosys can be performed on the logical network structure obtained by the module.

Keywords: reverse engineering, decompilation of transistor circuits, CMOS circuits, SPICE format, language Verilog, package Yosys

For citation. Cheremisinov D. I., Cheremisinova L. D. *Redesigning CMOS VLSI using Yosys synthesis tool*. Informatika [*Informatics*], 2025, vol. 22, no. 1, pp. 27−39 (In Russ.). DOI: 10.37661/1816-0301-2025-22-1-27-39.

Conflict of interest. The authors declare of no conflict of interest.

Введение. Современные сверхбольшие интегральные схемы (СБИС) содержат до миллиарда примитивных элементов на уровне транзисторов, и их сложность быстро увеличивается. Эффективным инструментом автоматизации тестирования и проектирования СБИС является обратное проектирование, или обратный инжиниринг (англ. hardware reverse engineering), схем на уровне транзисторов [1, 2]. Задача обратной инженерии инверсна задаче проектирования СБИС в смысле направления процесса преобразований. Она заключается в построении спецификации устройства путем анализа его аппаратной реализации в виде СБИС. Основным этапом обратного проектирования является декомпиляция плоской транзисторной схемы, которая состоит в извлечении описания уровня логических элементов [3, 4]. Полученное логическое описание устройства позволяет перепроектировать реализующую его транзисторную схему [2]. Перепроектирование, в отличие от оригинального проектирования, предполагает проектирование устройства на новой элементной основе для замены его существующей схемной реализации.

Процесс проектирования электронных устройств в любой системе автоматизированного проектирования (САПР) разбивается на ряд этапов преобразований от высокого уровня абстракции к низкому. Описание верхнего функционального уровня состоит из функциональных блоков, таких как триггеры, сумматоры и т. д., каждый из которых может содержать большое количество логических элементов. Логический уровень использует логические вентили как стандартные блоки, чтобы описать схемы. На транзисторном уровне схемы описываются в терминах транзисторов и их взаимосвязей.

Каждый этап проектирования требует своих специализированных инструментов. Программное обеспечение для автоматизации проектирования электронных устройств предлагают три основных мировых поставщика: Synopsys, Cadence Design Systems и Siemens EDA (ранее

Mentor Graphics, которая в 2017 г. перешла компании Siemens). Эти поставщики предлагают программные пакеты, позволяющие охватить весь спектр этапов проектирования интегральных схем: от синтеза по функционально-логическому описанию аппаратуры на языке HDL (Hardware Description Language – язык описания оборудования) до физического синтеза. Инструментальные средства САПР различаются набором не только предлагаемых проектных операций, элементных базисов и стилей проектирования, но и входных языков описания аппаратуры. Как правило, в качестве входных языков современных САПР используются языки VHDL и Verilog. Это требует наличия средств конвертации форматов представления разрабатываемых схем.

Разработка инструментов проектирования тесно связана с технологией изготовления микросхем. Свойства технологии конкретного завода полупроводников, такие как модели транзисторов, физические характеристики и правила проектирования, обычно фиксируются в описаниях, которые являются собственностью изготовителя программного обеспечения. Эти информационные материалы составляют набор PDK (Process Design Kit – комплект для процесса проектирования) для проектирования и процесса изготовления микросхем. Он разрабатывается обычно совместно изготовителем микросхем и поставщиком средств автоматизации проектирования, поэтому полупроводниковые предприятия выпускают PDK, которые совместимы с конкретным пакетом САПР.

Структура PDK, так же как и исходный код и структура промежуточных файлов данных САПР, является закрытой, а используемые технологии производства интегральных схем настолько сложны, что разработка средств автоматизации проектирования и продуктов, аналогичных существующим PDK, требует больших затрат времени квалифицированных разработчиков. Выходом из ситуации может быть использование свободно распространяемых программных пакетов САПР с открытым исходным кодом FOSS (Free and Open-Source Software – свободно доступное программное обеспечение с открытым исходным кодом). Эти пакеты в настоящее время находятся в стадии быстрой разработки благодаря проектам DARPA (Defense Advanced Research Projects Agency – агентство перспективных исследовательских проектов Министерства обороны США) и Google OpenROAD (Open Rapid Object Application Development – открытая быстрая разработка объектных приложений), которые содействуют разработке средств автоматизации проектирования микросхем с открытым исходным кодом. Проект OpenROAD [5] предлагает полный набор инструментов от высокоуровневого синтеза до генерации лейаута. Набор инструментов включает программные средства Yosys (Yosys Open Synthesis Suite – открытый пакет синтеза Yosys)[1] [6] для логического синтеза и OpenLane[2] [7] для синтеза топологии, которые ориентированы на выпуск заказных СБИС с технологией 130 нм (SkyWater 130 nm Open Source PDK[3]). В последнее время пакеты FOSS для проектирования интегральных схем совершенствуются благодаря усилиям энтузиастов из академических кругов и коммерческих организаций, которые пытаются упростить доступ и свободное использование набора инструментов для цифрового и аналогового проектирования микросхем.

Yosys представляет собой первое бесплатное программное обеспечение с открытым исходным кодом и открытой внутренней структурой представления данных, которое поддерживает подавляющее большинство синтезируемых функций языка Verilog HDL. Он создан как программный пакет с расширяемой структурой, его можно пополнять новыми программными компонентами. Текущее состояние Yosys позволяет обрабатывать, оптимизировать и синтезировать проекты на Verilog-2005, а также преобразовывать Verilog-описания в форматы BLIF (Berkeley Logic Interchange Format – формат обмена логическими данными Беркли), EDIF (Electronic Design Interchange Format – формат обмена проектами электронных устройств), RTL (Register Transfer Language – язык регистровых пересылок), Verilog, BTOR, SMT-LIB и др., доступные для использования в других САПР. Некоторые этапы процесса синтеза используют

[1]Yosys Open Synthesis Suite. – URL: https://yosyshq.net/yosys/ (date of access: 10.01.2025).

[2]OpenLane. The Open-Source Infrastructure Platform for Silicon Development. – URL: https://efabless.com/openlane (date of access: 10.01.2025).

[3]SkyWater SKY130 PDK. – 2020. – URL: https://skywater-pdk.readthedocs.io/en/main (date of access: 10.01.2025).

внешние инструменты, которые интегрированы в Yosys. Например, программные средства минимизации комбинационной логики и технологического отображения выполняются системой ABC[4] верификации и синтеза последовательностных схем, которая разработана группой Университета Беркли (Berkeley Logic Synthesis and Verification Group).

В настоящей работе рассматривается задача из области перепроектирования КМОП СБИС. Основным этапом обратного проектирования служит декомпиляция плоского нетлиста транзисторной схемы, заданной в формате SPICE (Simulation Program with Integrated Circuit Emphasis – программа моделирования с акцентом на интегральные схемы), которая заключается в извлечении описания на уровне логических элементов [3]. С помощью декомпиляции получается логическая сеть, представленная в виде иерархического SPICE-описания. Это описание может служить исходной спецификацией для выполнения оптимизации и синтеза СБИС, если представить его на используемых в САПР языках проектирования, таких как VHDL, Verilog и др. В работе предложены метод и его программная реализация для экспорта иерархического SPICE-описания логической сети в среду пакета синтеза Yosys[5] [6] прежде всего с целью преобразования SPICE-описания в другие форматы, а также для оптимизации и синтеза логической сети в рамках этого пакета.

Декомпиляция транзисторных схем. В статье [3] описывается программа декомпиляции КМОП-схем, которая предназначена для замены представления схемы на низком (транзисторном) уровне заданием на уровне логических элементов. Исходным объектом для программы служит плоский (одноуровневый) нетлист КМОП-схемы в формате SPICE, имя головной схемы и имена цепей питания. Результатом является иерархическое SPICE-описание, в которое включены модели всех идентифицированных КМОП-элементов. В программе распознаются логические КМОП-вентили статического стиля как самого распространенного, а также элементы на основе проходной логики – передаточные вентили и схемы на их основе, такие как мультиплексоры.

В процессе декомпиляции выполняются следующие основные этапы: выделение подсхем, представляющих передаточные вентили и схемы на их основе; распознавание подсхем, представляющих КМОП-вентили, и установление реализуемых ими функций; выделение групп функционально и топологически идентичных подсхем и формирование библиотеки элементов, из которых строится генерируемая логическая сеть; выделение подсхем с обратными связями; построение логической сети.

Полученная в результате декомпиляции логическая сеть представляется в виде иерархического описания в формате SPICE, в котором на нижних уровнях иерархии задаются модели подсхем, составляющих библиотеку логических элементов. Для каждого логического вентиля в процессе декомпиляции находится реализуемая им функция в скобочном представлении, которая приводится в текстовом виде в форме комментария в SPICE-описании соответствующей модели.

Полученная логическая сеть может быть выдана также в формате системы логических уравнений на языке SF иерархических функционально-структурных описаний дискретных устройств [8]. Описания на языке SF служат входными данными для разработанных в Объединенном институте проблем информатики НАН Беларуси систем логического проектирования функциональных блоков заказных КМОП СБИС [9]. В эти системы включены также средства [10] для конвертации SF-описаний дискретных устройств на язык VHDL. Для конвертации в другие языки проектирования (например, Verilog) предлагается применять средства пакета Yosys после преобразования SPICE-описания логической сети во внутреннее представление этого инструмента.

Задание логической сети в формате SPICE. В формате SPICE компоненты схемы описываются в виде моделей. Используются два основных типа моделей: модели устройств (device models) и модели подсхем (subcircuit models). Модели устройств, по сути, представляют собой примитивные компоненты, такие как транзисторы. Модель подсхемы определяет непримитивный элемент и обычно представляет собой схему, которая задается в виде блока в SPICE-описании, представляющем структуру соединений его элементов (примитивных и непримитив-

[4]ABC: A System for Sequential Synthesis and Verification / Berkeley Logic Synthesis and Verification Group. – URL: https://people.eecs.berkeley.edu/~alanmi/abc/ (date of access: 10.01.2025).

[5]URL: https://yosyshq.net/yosys/

ных). Тип модели используемого в SPICE-описании элемента задается первой буквой его имени. Имена примитивных элементов, являющихся МОП-транзисторами, должны начинаться с латинской буквы «M» (или «m»). Имена непримитивных элементов, представляемых подсхемами, начинаются с латинской буквы «X» (или «x»).

Электрическая схема в формате SPICE задается списком соединений ее элементов. В этом списке указаны связи между элементами (а точнее, между их выводами), которые осуществляются с помощью электрических цепей (nets). Цепи представляют собой связанный набор выводов элементов, на которые подается один и тот же сигнал, и каждый вывод элемента подключен ровно к одной цепи. Для всех элементов указываются связи всех их выводов путем задания имен связанных с ними цепей.

Описание транзистора начинается с его имени и содержит перечисление меток цепей, связанных с выводами стока, затвора, истока и подложки в заранее определенной последовательности. Общая форма описания связей униполярного транзистора в формате SPICE имеет вид

<name> <nd> <ng> <ns> <nb> <model-name>,

где name – имя транзистора в схеме (начиная с буквы «M»); nd, ng, ns и nb – идентификаторы цепей, связанных с выводами стока (drain), затвора (gate), истока (source) и подложки (substrate) соответственно; model-name – имя модели устройства (для *n*-МОП- и *p*-МОП-транзисторов это могут быть nmos и pmos).

В качестве примера в листинге 1 приведено плоское описание транзисторной схемы (рис. 1) в формате SPICE, представляющей полный одноразрядный сумматор из обзора[6].

Листинг 1. *SPICE-описание полного одноразрядного сумматора*

```
.SUBCKT adder a b cin cout sum vdd gnd
M0 vdd a 1 vdd p                    M14 vdd cin 6 vdd p
M1 vdd b 1 vdd p                    M15 6 a 7 vdd p
M2 1 b 2 vdd p                      M16 7 b 8 vdd p
M3 2 a 3 vdd p                      M17 8 cin 9 vdd p
M4 1 cin 3 vdd p                    M18 6 3 9 vdd p
M5 3 a 4 gnd n                      M19 vdd 9 sum vdd p
M6 4 b gnd gnd n                    M20 9 a 10 gnd n
M7 3 cin 5 gnd n                    M21 10 b 11 gnd n
M8 5 a gnd gnd n                    M22 11 cin gnd gnd n
M9 5 b gnd gnd n                    M23 12 a gnd gnd n
M10 cout 3 gnd gnd n                M24 12 b gnd gnd n
M11 vdd 3 cout vdd p                M25 12 cin gnd gnd n
M12 vdd a 6 vdd p                   M26 9 3 12 gnd n
M13 vdd b 6 vdd p                   M27 sum 9 gnd gnd n
                                    .ENDS
```

Общая форма описания связей непримитивного элемента, модель которого представляется подсхемой с *n* выводами, в формате SPICE имеет вид

<name> <P1> < P2> …< Pn> <model-name>,

где name – имя элемента в схеме (начиная с буквы «X»); «P1», «P2»… «Pn» – идентификаторы цепей, связанных с выводами элемента; model-name – имя модели элемента.

В формате SPICE выводы элементов не делятся на входные и выходные, но это разделение существенно для представления логических схем. В программе декомпиляции принято следующее соглашение: при описании моделей КМОП-элементов, которые являются (*n*, 1)-полюсниками, последний вывод всегда отождествляется с выходным полюсом элемента и именуется как «Y». Кроме того, распознанная при декомпиляции функция, реализуемая элементом, приводится в описании его SPICE-модели в виде текстового комментария, в котором знак инверсии опущен.

[6]CMOS Binary Full Adder. A Survey of Possible Implementations. – URL: http://web.engr.uky.edu/~elias/projects/10.pdf (date of access: 10.01.2025).

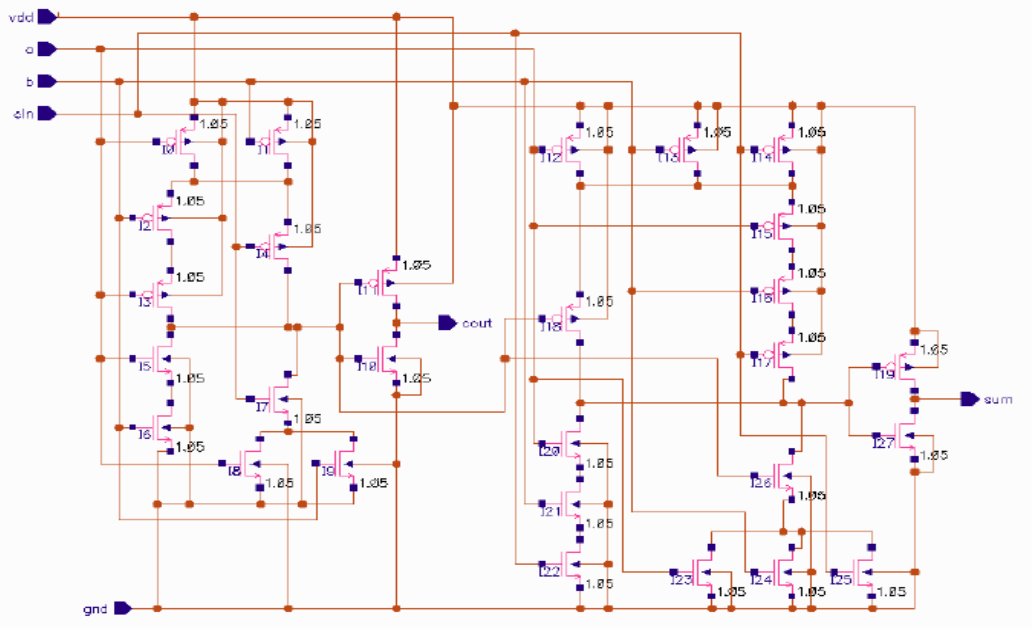


Рис. 1. Транзисторная схема полного одноразрядного сумматора
Fig. 1. Transistor circuit of a full single-bit adder

Результат декомпиляции приведенной выше транзисторной схемы одноразрядного сумматора представлен иерархическим SPICE-описанием (листинг 2), в которое включены модели трех распознанных подсхем G0, G1 и G2, представляющих КМОП-вентили. Эти модели и составляют извлеченную при декомпиляции библиотеку вентилей. Вентили XM2I1 и XM2I2 являются инверторами, они топологически изоморфны, их модель имеет имя G2. Остальные два вентиля XM0I1 и XM1I1, модели которых имеют имена G0 и G1, реализуют функции $\overline{ABC \vee G(D \vee E \vee F)}$ и $\overline{AB \vee C(D \vee E)}$.

Листинг 2. *Иерархическое SPICE-описание одноразрядного сумматора, полученное после выделения логической сети*

```
* SPICE deck for cell adder_gen
.GLOBAL  vdd  gnd
.SUBCKT G0 A B C D E F G Y
*((A AND B AND C)OR(G AND (D OR E OR F)))
M1 Y A 2 gnd n
M2 2 B 3 gnd n
M3 3 C gnd gnd n
M4 5 D gnd gnd n
M5 5 E gnd gnd n
M6 5 F gnd gnd n
M7 Y G 5 gnd n
M8 vdd A 7 vdd p
M9 vdd B 7 vdd p
M10 vdd C 7 vdd p
M11 7 A 8 vdd p
M12 8 E 9 vdd p
M13 9 F Y vdd p
M14 7 G Y vdd p
.ENDS
.SUBCKT G1 A B C D E Y
*((A AND B) OR (C AND (D OR E)))
M1 Y A 2 gnd n
M2 2 B gnd gnd n
M3 Y C 4 gnd n
M4 4 D gnd gnd n
M5 4 E gnd gnd n
M6 vdd A 6 vdd p
M7 vdd B 6 vdd p
M8 6 E 7 vdd p
M9 7 A Y vdd p
M10 6 C Y vdd p
.ENDS
.SUBCKT G2 A Y
* A
M1 Y A gnd gnd n
M2 vdd A Y vdd p
.ENDS
.SUBCKT C0 P0 P1 P2 O3 O4
XG0M0I1 P0 P1 P2 P0 P1 P2 1 2 G0
XG1M1I1 P0 P1 P2 P0 P1 1 G1
XG2M2I1 1 O3 G2
XG2M2I2 2 O4 G2
.ENDS
.SUBCKT adder_gen a b cin cout sum
XC0 a b cin cout sum C0
.ENDS
```

Найденные при декомпиляции логические сети (их может быть несколько даже для одного описания [11]) выделяются как отдельные модели с именами, начинающимися с символа «C». Для отображения информации о типах портов схемы (формат SPICE не содержит средств для их указания) в списке выводов входные полюсы принято именовать, начиная с символа «P», а выходные полюсы – с символа «O».

Для рассматриваемой схемы сумматора получилась одна логическая сеть (листинг 2, рис. 2) с именем C0, которая имеет пять выводов, три из которых (P0, P1 и P2) являются входами, два (O3 и O4) – выходами. Соответствие параметров логической сети C0 параметрам исходного SPICE-описания транзисторной схемы adder задается экземпляром XC0 этой сети в сгенерированном описании adder_gen (см. листинг 2).

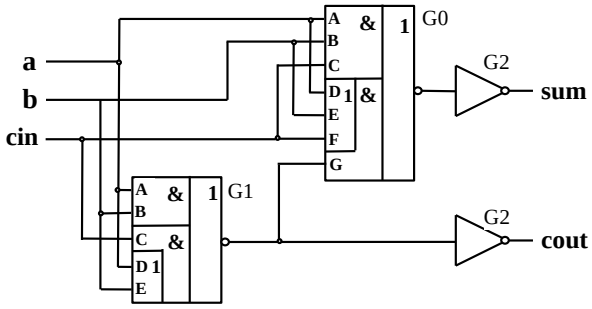


Рис. 2. Логическая сеть одноразрядного сумматора
Fig. 2. Logical network of a single-bit adder

Структура и представление данных в Yosys. Программный пакет Yosys охватывает три уровня проектирования: поведенческий, синтез на уровне регистровых передач и логический. Он принимает описание проекта на поведенческом уровне (на языке Verilog) в качестве входных данных и генерирует в качестве выходных данных описание логической сети уровня RTL логических вентилей в разных форматах. Кроме того, Yosys позволяет отображать проекты на языке Verilog в базис библиотечных ячеек заказных БИС в формате Liberty и в базис ПЛИС Xilinx седьмой серии. Для реализации логической оптимизации и отображения в указанные базисы используется встроенный в Yosys программный инструмент ABC[7].

В структурном плане Yosys представляет собой модульную программу, которая управляется с помощью скриптов. Каждая команда в скрипте относится к одному из трех классов: фронтенд (frontend), проход (pass) или бэкенд (backend). Команда типа фронтенд позволяет пользователю ввести в Yosys описание объекта проектирования, а команда типа бэкенд – получить файл с результатом. К типу проход (pass) относятся команды, которые выполняют операции над внутренним представлением объекта проектирования: анализ или преобразование (оптимизацию, синтез, перевод в другой формат). Процесс работы с новым объектом проектирования начинается с одной из команд типа фронтенд, далее команды могут выполняться в любом произвольном порядке, так как они работают с одной и той же структурой данных, которая порождена командой типа фронтенд (и может быть дополнена при выполнении команд типа pass). Команды типа бэкенд используются для записи результата преобразования описания объекта проектирования в некотором формате в выходной файл.

С функциональной точки зрения программа Yosys состоит из ядра и обрабатывающих модулей, выполнение которых запускается командами встроенного интерпретатора командной строки. Ядро Yosys представляет собой совокупность структур данных и методов, которые применяются для представления логической сети и манипуляций с ней. Обрабатывающие модули взаимодействуют через интерфейс, являющийся промежуточным объектом в формате RTLIL (Register Transfer Level Intermediate Language – промежуточный язык регистровых пересылок). RTLIL-структура, представляющая проект, используется всеми проходами Yosys в качестве и входных, и выходных данных. Проходы выполняют все изменения на месте полученной на

[7]URL: https://people.eecs.berkeley.edu/~alanmi/abc/

входе структуры проекта, передавая на выход всю структуру целиком. Поэтому не зависящие друг от друга проходы (и команды типа backend) можно выполнять в произвольном порядке.

Представление проекта в формате RTLIL, по сути, является заданием списка соединений схемы со следующей особенностью: используется внутренняя для Yosys библиотека вентилей, которые имеют фиксированные функции. Номенклатура вентилей библиотеки соответствует логическим операторам языка Verilog (функции AND, OR, XOR, NOT, NAND, NOR). Однако имеется также и вентиль LUT (Lookup Table), реализуемая им булева функция задается в векторном виде. В зависимости от своей настройки вентиль LUT может реализовать любую необходимую функцию от заданного числа входных сигналов. Кроме логических вентилей библиотека содержит также мультиплексор.

Структура данных RTLIL представляет собой класс языка C++. Корневым объектом структуры данных RTLIL является класс RTLIL::Design, который представляет собой обрабатываемую логическую сеть. В памяти Yosys всегда есть только один активный объект RTLIL::Design – текущий проект. Данные в него добавляются фронтендами (иногда и проходами), бэкенды преобразуют их в экспортируемые форматы. По умолчанию, именно текущий проект трансформируется обрабатывающими модулями, запускаемыми командами языка-интерпретатора командной строки.

Цель последовательности трансформаций, задаваемых проходами, – преобразование вводимой логической сети в состояние, в котором ее функциональность реализуется вентилями из заданной библиотеки и проводами для связей вентилей друг с другом. Класс RTLIL::Design является верхним слоем иерархии объектов, представляющих ядро, и может содержать несколько объектов класса RTLIL::Module (для сложных проектов), которые соответствуют модулям в Verilog. Объект RTLIL::Design служит контейнером для объектов RTLIL::Module.

Модуль RTLIL::Module, в свою очередь, включает в себя объекты трех основных типов:

– RTLIL::Cell и RTLIL::Wire, представляющие списки ячеек схемы и их связей;

– RTLIL::Process, описывающие конструкции типа if-then-else;

– RTLIL::Memory, представляющие адресуемые ячейки памяти.

При выполнении проходов синтеза объекты RTLIL::Process и RTLIL::Memory заменяются объектами RTLIL::Cell и RTLIL::Wire, которые задают списки соединений ячеек схемы.

В формате RTLIL все идентификаторы (модулей, портов, сигналов, ячеек и др.) начинаются с символа «\», если они заданы в исходном описании проекта, или с символа «$», если они сгенерированы автоматически. При обработке объекта, заданного RTLIL, бэкендом последние идентификаторы преобразуются так, чтобы не конфликтовать с идентификаторами первого типа.

Экспорт логической сети в формате SPICE в Yosys. Yosys спроектирован как расширяемый комплекс программ. Его ядро можно дополнять командами, реализующими новые алгоритмы синтеза и обработки данных, представляемых на внутреннем языке RTLIL. Для экспорта в ядро Yosys логической сети в формате SPICE был разработан фронтенд, который принимает иерархическое SPICE-описание логической схемы и переводит его во внутреннее представление Yosys в формате RTLIL.

Фронтенд реализован программой с именем my_cmD на языке C++ с использованием классов программного пакета Yosys. Разработанная программа принимает текстовое описание логической схемы в формате SPICE и генерирует соответствующий класс RTLIL::Design, состоящий из одного класса RTLIL::Module. Над полученной таким образом структурой могут быть выполнены все доступные в Yosys преобразования.

Получаемое представление схемы объектом RTLIL::Module состоит из объектов RTLIL::Cell и RTLIL::Wire, которые составляют структуру схемы со следующей особенностью, диктуемой требованиями Yosys: функциональность схемы реализуется ячейками только из заданной внутренней для Yosys библиотеки ячеек и проводами для соединения этих ячеек друг с другом.

Обработка исходного SPICE-описания разработанным фронтендом начинается с анализа его текстового представления, в процессе которого формируется внутреннее представление структуры SPICE-описания: иерархия хеш-таблиц для моделей элементов схемы (в том числе и мо-

дели самой схемы) с указанием их имен, типов, идентификаторов их входных и выходных полюсов, реализуемых функций.

В процессе анализа для модели анализируемой логической сети строится также представление, отображающее связи ее элементов, в виде помеченного двудольного орграфа $G = (V_1, V_2, E)$, $V_1 \cap V_2 = \varnothing$. В нем вершины из первой доли V_1 графа соответствуют выводам экземпляров элементов, а вершинам из V_2 ставятся в соответствие цепи и порты схемы в целом. Каждая из дуг $e \in E$ графа G связывает вершины из разных множеств V_1 и V_2. Вершины графа из множества V_2 помечаются именами цепей и портов схемы, а вершины из V_1 имеют метки, которые идентифицируют выводы экземпляров элементов и состоят из имени экземпляра элемента и имени вывода модели этого элемента. Двудольный граф $G = (V_1, V_2, E)$, представляющий модель схемы в формате SPICE, является разреженным, и степени всех вершин в доле V_1 выводов элементов равны единице. Для задания связей такого графа логично группирование дуг, входящих и исходящих из вершин множества V_1, которые соответствуют выводам одного и того же экземпляра элемента сети. В процессе анализа SPICE-описания для каждого экземпляра элемента сети задается список связей его выводов, приведенных в том порядке, в котором они упоминаются в описании связей модели этого элемента в формате SPICE.

Для приведенного выше SPICE-описания множества V_1 и V_2 графа G имеют 18 и 7 вершин, а связи графа задаются четырьмя списками ссылок на описания выводов элементов XG0M0I1, XG1M1I1, XG2M2I1 и XG2M2I2.

Полученная иерархия хеш-таблиц полностью отражает структуру схемы, представленной SPICE-описанием. По этому представлению генерируются списки объектов RTLIL::Cell и RTLIL::Wire, которые порождаются множествами V_1 и V_2 вершин графа $G = (V_1, V_2, E)$. При генерации объектов RTLIL::Wire (провода) для них указываются идентификаторы, ширина (равная единице в рассматриваемом случае) и направление (in/out/inout).

При генерации объектов RTLIL::Cell для каждой ячейки схемы указывается идентификатор соответствующего экземпляра элемента и его тип, идентификаторы и типы портов (входы, выходы), а также реализуемая ячейкой функция. Реализуемая функция находится по ее текстовому описанию, выбираемому из внутреннего представления SPICE-описания. Если это простая одновыходная функция (типа NAND, NOR и др.), которая реализуется одним из вентилей внутренней для Yosys библиотеки, то в RTLIL::Cell ячейка помечается типом соответствующего библиотечного элемента. В случае более сложной функции ячейка помечается типом вентиля LUT (Lookup Table), а реализуемая булева функция от n аргументов задается в векторном виде 2^n-разрядным вектором ее значений. Связи между портами ячеек и цепями задаются с помощью пар объектов RTLIL::SigSpec (или объектом типа RTLIL::SigSig).

Разработанная программа my_cmD экспорта SPICE-описания была интегрирована в среду Yosys в качестве одного из фронтендов. При обращении к ней во время работы программного интерпретатора Yosys в командной строке указываются два параметра: <Par1> и <Par2>, где Par1 задает путь доступа к файлу со SPICE-описанием, а Par2 – имя модели анализируемой логической сети в этом описании.

Пример выполнения разработанного обрабатывающего модуля в среде Yosys. Работа программного модуля my_cmd в среде Yosys демонстрируется на примере импорта иерархического SPICE-описания логической сети C0 (листинг 2) из файла d:\abdata23\adder_gen_ier.sp. В Yosys это описание преобразуется с помощью команд типа write в описания на языках Verilog, BLIF и EDIF. На листинге 3 приведено состояние консольного окна программного интерпретатора Yosys при выполнении соответствующих команд.

Листинг 3. *Консольное окно интерпретатора Yosys*

```
yosys> my_cmd d:\2025\adder\adder_gen_ier.sp C0
Arguments to my_cmd:
  my_cmd
  d:\2025\adder\adder_gen_ier.sp
  C0
Read input file d:\2025\adder\adder_gen_ier.sp
```

```
Reading is OK
Modules in current design:
  C0 (7 wires, 4 cells)
yosys> write_verilog d:\abdata23\adder.v
1. Executing Verilog backend.
Dumping module `\C0'.
yosys> write_blif d:\abdata23\adder.blif
2. Executing BLIF backend.
yosys> write_edif d:\abdata23\adder.edif
3. Executing EDIF backend.
```

На листингах 4 и 5 показаны полученные представления на языках Verilog и BLIF для импортированного в Yosys описания одноразрядного сумматора (описание в формате EDIF не приведено в силу его громоздкости). В Verilog-описании функции (вид этих функций от семи и пяти аргументов приведен в скобках в виде комментариев) первых двух ячеек заданы в виде векторов длиной 2^7 и 2^5, две другие ячейки представляют собой инверторы. В BLIF-описании строчки векторов значений аргументов функций с идентификаторами «\2» и «\1» разбиты на пять групп.

Листинг 4. *Описание логической сети сумматора на языке Verilog*

```
/* Generated by Yosys 0.9 (git sha1 1979e0b1, Visual Studio) */
module C0(Ca, Cb, Ccin, Ccout, Csum);
  wire \1 ;
  wire \2 ;
  input Ca;
  input Cb;
  input Ccin;
  output Ccout;
  output Csum;
  assign \2  = 128'h00005557555755575557555755575557 >> (* src = "Y=((A * B * C) +
(G * (D + E + F)))" *) { \1 , Ccin, Cb, Ca, Ccin, Cb, Ca };
  assign \1  = 32'd2039583 >> (* src = "Y=((A * B) + (C * (D + E)))" *) { Cb, Ca, Ccin,
Cb, Ca };
  assign Ccout = ~\1 ;
  assign Csum = ~\2 ;
endmodule
```

Листинг 5. *Описание логической сети сумматора в формате BLIF*

```
# Generated by Yosys 0.9 (git sha1 1979e0b1, Visual Studio)
.model C0
.inputs Ca Cb Ccin
.outputs Ccout Csum
.names $false
.names $true
1
.names $undef
.names \1 Ccin Cb Ca Ccin Cb Ca \2
0000000 1    0010110 1    0101110 1    1000100 1    1011100 1
0000001 1    0011000 1    0110000 1    1000110 1    1011110 1
0000010 1    0011010 1    0110001 1    1001000 1    1100000 1
0000100 1    0011100 1    0110010 1    1001010 1    1100001 1
0000110 1    0011110 1    0110100 1    1001100 1    1100010 1
0001000 1    0100000 1    0110110 1    1001110 1    1100100 1
0001010 1    0100001 1    0111000 1    1010000 1    1100110 1
0001100 1    0100010 1    0111010 1    1010001 1    1101000 1
0001110 1    0100100 1    0111100 1    1010010 1    1101010 1
```

```
0010000 1     0100110 1     0111110 1     1010100 1     1101100 1
0010001 1     0101000 1     1000000 1     1010110 1     1101110 1
0010010 1     0101010 1     1000001 1     1011000 1
0010100 1     0101100 1     1000010 1     1011010 1
.names Cb Ca Ccin Cb Ca \1
00000 1       00011 1       01001 1       01100 1       10010 1
00001 1       00100 1       01010 1       10000 1       10011 1
00010 1       01000 1       01011 1       10001 1       10100 1
.names \1 Ccout
0 1
.names \2 Csum
0 1
.end
```

BLIF является входным форматом пакета оптимизации синтеза ABC[8], который интегрирован в Yosys. Операции синтеза можно производить в среде Yosys для Verilog-описаний, а также в среде ABC для BLIF-описаний. Например, с помощью команд пакета ABC

abc 01> read_blif d:\2025\adder\adder.blif

abc 02> write_eqn d:\2025\adder\adder.eqn

было получено описание сумматора в виде логических уравнений (листинг 6).

Листинг 6. *Описание логической сети сумматора в виде логических уравнений*

```
# Equations for "C0" written by ABC on Thu Jan 30 15:59:29 2025
INORDER = Ca Cb Ccin;
OUTORDER = Ccout Csum;
new_\2_ = ((!new_\1_ + (new_\1_ * !Ccin)) * (!Ca + (!Ca * !Ccin * !Cb *
Ca))) + (new_\1_ * Ccin * !Cb * (!Ca + (!Ca * !Ccin * !Cb * Ca)));
new_\1_ = (!Cb + (Cb * !Ca)) * (!Ccin + (Ccin * !Cb * !Ca));
Ccout = !new_\1_;
Csum = !new_\2_;
```

Заключение. Разработана программа преобразования иерархического описания логической сети в формате SPICE в описание на внутреннем языке RTLIL инструмента синтеза Yosys. Программа оформлена в виде фронтенда Yosys. Расширенный пакет Yosys позволяет выполнять моделирование и перепроектирование схемы до уровня лейаута для схем, полученных путем декомпиляции СБИС на уровне транзисторов. Предлагаемая разработка полезна не только для исследовательского и академического применения, когда требуются модификации инструментов и описаний, но и для проектирования и перепроектирования СБИС промышленными САПР.

Вклад авторов. *Д. И. Черемисинов* разработал метод и программные средства экспорта иерархического SPICE-описания логической сети в среду пакета синтеза Yosys. *Л. Д. Черемисинова* разработала метод анализа иерархического SPICE-описания и подготовила текст статьи.

Список использованных источников

1. Baker, R. J. CMOS Circuit Design, Layout, and Simulation / R. J. Baker. – Third ed. – Wiley-IEEE Press, 2010. – 1214 p.

2. Hunt, V. D. Reengineering: Leveraging the Power of Integrated Product Development / V. D. Hunt. – Wiley, 1993. – 283 p.

3. Черемисинов, Д. И. Извлечение сети логических элементов из КМОП-схемы транзисторного уровня / Д. И. Черемисинов, Л. Д. Черемисинова // Микроэлектроника. – 2019. – Т. 48, № 3. – С. 224–234. – DOI: 10.1134/S0544126919030037.

[8]URL: https://people.eecs.berkeley.edu/~alanmi/abc/

4. Yang, L. FROSTY: A program for fast extraction of high-level structural representation from circuit description for industrial CMOS circuits / L. Yang, C.-J. R. Shi // Integration the VLSI Journal. – 2006. – Vol. 39, no 4. – P. 311–339.

5. Ajayi, T. Toward an open-source digital flow: First learnings from the OpenROAD project / T. Ajayi, V. A. Chhabria, M. Fogaça [et al.] // Proc. of the 56th Annual Design Automation Conf. (DAC '19), Las Vegas, NV, USA, 02–06 June 2019. – Las Vegas, 2019. – Article 76. – P. 1–4.

6. Wolf, C. Yosys – a free verilog synthesis suite / C. Wolf, J. Glaser, J. Kepler // Proc. of the 21st Austrian Workshop on Microelectronics (Austrochip 2013), Linz, Austria, 10 Oct. 2013. – Linz, 2013. – URL: https://yosyshq.net/yosys/files/yosys-austrochip2013.pdf (date of access: 10.01.2025).

7. Ghazy, A. A. OpenLANE: The Open-Source Digital ASIC Implementation Flow / A. A. Ghazy, M. Shalan. – URL: https://woset-workshop.github.io/PDFs/2020/a21.pdf (date of access: 10.01.2025).

8. Бибило, П. Н. Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний / П. Н. Бибило, В. И. Романов. – Минск : Беларус. навука, 2011. – 279 с.

9. Система логического проектирования функциональных блоков заказных КМОП СБИС с пониженным энергопотреблением / П. Н. Бибило, Н. А. Авдеев, С. Н. Кардаш [и др.] // Микроэлектроника. – 2018. – Т. 47, № 1. – С. 71–87.

10. Черемисинов, Д. И. Анализ и преобразование структурных описаний СБИС / Д. И. Черемисинов. – Минск : Беларус. навука, 2006. – 275 с.

11. Черемисинов, Д. И. Извлечение логических сетей при декомпиляции описаний КМОП-схем на уровне транзисторов / Д. И. Черемисинов, Л. Д. Черемисинова // Информатика. – 2024. − Т. 21, № 3. – С. 23–38.

References

1. Baker R. J. *CMOS Circuit Design, Layout, and Simulation*, third edition. Wiley-IEEE Press, 2010, 1214 p.

2. Hunt V. D. *Reengineering: Leveraging the Power of Integrated Product Development*. Wiley, 1993, 283 p.

3. Cheremisinov D. I., Cheremisinova L. D. *Extracting a logic gate network from a transistor-level CMOS circuit.* Mikrojelektronika [*Russian Microelectronics*], 2019, vol. 48, no. 3, pp. 224–234 (In Russ.). DOI: 10.1134/S0544126919030037.

4. Yang L., Shi C.-J. R. FROSTY: A program for fast extraction of high-level structural representation from circuit description for industrial CMOS circuits. *Integration the VLSI Journal*, 2006, vol. 39, no 4, pp. 311–339.

5. Ajayi T., Chhabria V. A., Fogaça M., Hashemi S., Hosny A., …, Xu B. Toward an open-source digital flow: First learnings from the OpenROAD project. *Proceedings of the 56th Annual Design Automation Conference (DAC '19), Las Vegas, NV, USA, 02–06 June 2019.* Las Vegas, 2019, article 76, pp. 1–4.

6. Wolf C., Glaser J., Kepler J. Yosys – a free verilog synthesis suite. *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip 2013), Linz, Austria, 10 October 2013*. Linz, 2013. Available at: https://yosyshq.net/yosys/files/yosys-austrochip2013.pdf (accessed 10.01.2025).

7. Ghazy A. A., Shalan M. *OpenLANE: The Open-Source Digital ASIC Implementation Flow*. Available at: https://woset-workshop.github.io/PDFs/2020/a21.pdf (accessed 10.01.2025).

8. Bibilo P. N., Romanov V. I. Logicheskoye proyektirovaniye diskretnykh ustroystv s ispol'zovaniyem produktsionno-freymovoy modeli predstavleniya znaniy. *Logical Design of Discrete Devices Using a Production-Frame Model of Knowledge Representation*. Minsk, Belaruskaja navuka, 2011, 279 p. (In Russ.).

9. Bibilo P. N., Avdeev N. A., Kardash S. N., Kirienko N. A., Lankevich Ju. Ju., …, Cheremisinova L. D. *A system for logical design of custom CMOS VLSI functional blocks with reduced power consumption*. Mikrojelektronika [*Russian Microelectronics*], 2018, vol. 47, no. 1, pp. 65–81 (In Russ.).

10. Cheremisinov D. I. Analiz i preobrazovaniye strukturnykh opisaniy SBIS. *Analysis and Transformation of VLSI Structural Descriptions.* Minsk, Belaruskaja navuka, 2006, 275 p. (In Russ.).

11. Cheremisinov D. I., Cheremisinova L. D. *Extraction of logical networks during decompiling transistor-level CMOS circuit descriptions.* Informatika [Informatics], 2024, vol. 21, no. 3, pp. 23−38 (In Russ.). DOI: 10.37661/1816-0301-2024-21-3-23-38.

Информация об авторах

Черемисинов Дмитрий Иванович, кандидат технических наук, доцент, ведущий научный сотрудник, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: cher@newman.bas-net.by

Черемисинова Людмила Дмитриевна, доктор технических наук, профессор, главный научный сотрудник, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: cld@newman.bas-net.by

Information about the authors

Dmitry I. Cheremisinov, Ph. D. (Eng.), Assoc. Prof., Leading Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: cher@newman.bas-net.by

Ljudmila D. Cheremisinova, D. Sc. (Eng.), Prof., Chief Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: cld@newman.bas-net.by

УДК 519.714.5
DOI: 10.37661/1816-0301-2025-22-1-40-65

Оригинальная статья
Original Article

Дизъюнктивные и конъюнктивные разложения не полностью определенных булевых функций в бинарной диаграмме решений

П. Н. Бибило

Объединенный институт проблем информатики
Национальной академии наук Беларуси,
ул. Сурганова, 6, Минск, 220012, Беларусь
E-mail: bibilo@newman.bas-net.by

Аннотация

Цели. Рассматриваются задачи минимизации числа функций – кофакторов (подфункций) разложения Шеннона, находящихся на одном уровне бинарной диаграммы решений (Binary Decision Diagram, BDD), которая представляет систему не полностью определенных (частичных) булевых функций. Для сокращения числа функций предлагается находить подмножество таких функций, которые могут быть выражены в виде алгебраических разложений – дизъюнкций либо конъюнкций других доопределенных частичных функций. При этом ориентированный граф вхождений функций в разложения не должен содержать контуров.

Методы. Нахождение дизъюнктивных и конъюнктивных разложений требует поиска соответствующих доопределений частичных функций. Задача определения наибольшего числа раздельных алгебраических разложений сводится к нахождению взвешенного строчного покрытия булевой матрицы вхождений функций системы в отдельные разложения. Задача нахождения согласованных доопределений частичных функций для различного вида совместных разложений сводится к составлению и решению логических уравнений.

Результаты. Показано, что составляемые логические уравнения легко преобразуются к конъюнктивной нормальной форме (КНФ), а нахождение корней таких уравнений сведено к задаче выполнимости булевой формулы, представленной в виде КНФ, для решения которой известны эффективные методы и программы.

Заключение. Предложенные алгоритмы могут быть обобщены для других видов алгебраических разложений, когда кроме логических операций дизъюнкции и конъюнкции могут использоваться отрицания данных операций. Применение предложенных алгоритмов и уже известных алгоритмов минимизации многоуровневых BDD-представлений систем частичных функций позволяет получать лучшие результаты технологически независимой логической оптимизации – начального этапа синтеза логических схем.

Ключевые слова: система частичных булевых функций, доопределение частичной булевой функции, бинарная диаграмма решений, алгебраические разложения булевых функций, разложение Шеннона, проблема выполнимости

Для цитирования. Бибило, П. Н. Дизъюнктивные и конъюнктивные разложения не полностью определенных булевых функций в бинарной диаграмме решений / П. Н. Бибило // Информатика. – 2025. − Т. 22, № 1. – С. 40–65. – DOI: 10.37661/1816-0301-2025-22-1-40-65.

Конфликт интересов. Автор заявляет об отсутствии конфликта интересов.

Поступила в редакцию | Received 06.02.2025
Подписана в печать | Accepted 21.02.2025
Опубликована | Published 31.03.2025

Disjunctive and conjunctive decompositions of incompletely defined Boolean functions in a Binary Decision Diagram

Petr N. Bibilo

The United Institute of Informatics Problems
of the National Academy of Sciences of Belarus,
st. Surganova, 6, Minsk, 220012, Belarus
E-mail: bibilo@newman.bas-net.by

Abstract

Objectives. The problems of minimizing the number of cofactors (subfunctions) of the Shannon expansions located at the same level of the BDD, representing a system of incompletely defined (partial) Boolean functions, are considered. To reduce the number of functions, it is proposed to find a subset of such functions that can be expressed as algebraic decompositions of disjunctions or conjunctions of other predefined partial functions, while the directed graph of function occurrences in decompositions should not contain contours.

Methods. Finding disjunctive and conjunctive decompositions requires searching for appropriate additional definitions of partial functions. Finding the largest number of separate algebraic decompositions reduces to the problem of finding a weighted row cover of a Boolean matrix of occurrences of system functions in separate decompositions. The task of finding consistent predefinitions of partial functions for various types of joint decompositions is reduced to the formulation and solution of logical equations.

Results. It is shown that the constructed logical equations can be easily transformed to a conjunctive normal form (CNF), and finding the roots of such equations is reduced to the problem of satisfiability of a Boolean formula presented in the form of CNF, for which effective methods and programs are known.

Conclusion. The proposed algorithms can be generalized to other types of algebraic decompositions, when, in addition to the logical operations of disjunction and conjunction, negations of these operations can be used. The application of the proposed algorithms and already known algorithms for minimizing multilevel BDD representations of partial function systems allows us to obtain the best results of technologically independent logical optimization, the initial stage of logic circuit synthesis.

Keywords: system of partial Boolean functions, additional definition of a partial Boolean function, Binary Decision Diagram, algebraic decompositions of Boolean functions, Shannon expansion, SAT-problem

For citation. Bibilo P. N. *Disjunctive and conjunctive decompositions of incompletely defined Boolean functions in a Binary Decision Diagram.* Informatika [*Informatics*], 2025, vol. 22, no. 1, pp. 40–65 (In Russ.). DOI: 10.37661/1816-0301-2025-22-1-40-65.

Conflict of interest. The author declares of no conflict of interest.

Введение. BDD строятся на основе разложений Шеннона булевых функций. Они нашли широкое применение в качестве представлений систем полностью определенных булевых функций на этапе технологически независимой оптимизации [1], после которого выполняется второй этап синтеза логической схемы – технологическое отображение в требуемый базис логических элементов. Использование дизъюнктивных и конъюнктивных разложений подфункций (кофакторов), получаемых в результате применения разложений Шеннона, которые представляют системы полностью определенных булевых функций, оказалось полезной дополнительной процедурой, позволяющей уменьшать площадь функциональных блоков цифровых заказных сверхбольших интегральных схем (СБИС) [2, 3].

В настоящей работе рассматриваются задачи нахождения разложений подмножества не полностью определенных (частичных) булевых функций исходной системы в виде алгебраических (двухоперандных) дизъюнкций либо конъюнкций других частичных функций этой системы, названных базисными функциями. В качестве функций системы рассматриваются кофакторы одного уровня графа BDD [3]. При решении данной задачи находится максимальное по мощности подмножество функций системы, представимых в виде дизъюнкций либо конъюнкций. При этом ориентированный граф вхождений функций в разложения, т. е. граф зависимо-

сти разлагаемых функций от базисных (и уже разложенных), не должен содержать контуров. Нахождение таких разложений требует для базисных функций замен некоторых неопределенных значений функций определенными – нулями либо единицами, чтобы разложение могло быть выполнено. Использование одной и той же частичной функции для разложений нескольких функций системы является более сложной задачей, так как требует согласованных доопределений всех функций, участвующих в совместном разложении. Проверка существования согласованных доопределений сводится к составлению и решению логического уравнения. Корни уравнения дают требуемые замены неопределенных значений определенными для каждой из частичных функций, участвующих в совместном разложении. Получаемые уравнения легко преобразуются к форме КНФ. Широко известная задача нахождения хотя бы одного корня такого уравнения называется в литературе «выполнимость КНФ» [4]. Для решения задачи выполнимости КНФ известны эффективные методы и программы, называемые SAT-solvers [5]. Они позволяют решать задачи большой размерности, когда число переменных достигает нескольких миллионов. Предлагаемые эвристические алгоритмы нахождения алгебраических разложений ориентированы на решение практических задач большой размерности, так как на одном уровне BDD может быть достаточно много (десятки и сотни) частичных функций, сокращение числа которых возможно выполнить предлагаемыми алгоритмами.

Основные определения. *Булевыми* называются двоичные (0, 1) функции $f(\boldsymbol{x})=f(x_1, x_2,\ldots, x_n)$ двоичных (булевых) переменных $x_1, x_2, ..., x_n$. Пусть V^x – булево пространство, построенное над переменными булева вектора $\boldsymbol{x}=(x_1, x_2,\ldots, x_n)$. Элементами этого пространства являются n-компонентные наборы (векторы) $\boldsymbol{x}^*$ нулей и единиц. Булева функция, значения 0, 1 которой определены на всех элементах $\boldsymbol{x}^* \in V^x$, называется *полностью определенной*. Если же на некоторых элементах булева пространства V^x значения функции не определены, то такая функция будет *не полностью определенной*, или *частичной*. Частичная булева функция принимает единичное значение на элементах $\boldsymbol{x}^*$ подмножества M_f^1 булева пространства V^x и нулевое значение на элементах подмножества M_f^0. На всех остальных элементах пространства V^x, образующих подмножество M_f^- пространства V^x, значения частичной функции не определены. Подмножество M_f^- будем называть также областью *неопределенных значений* частичной функции. Неопределенное значение функции обозначается символом «–». Очевидно, что $M_f^1 \cap M_f^0 = \varnothing$, $M_f^1 \cap M_f^- = \varnothing$, $M_f^0 \cap M_f^- = \varnothing$, $V^x = M_f^0 \cup M_f^1 \cup M_f^-$.

Частичные функции $f_1(\boldsymbol{x})$ и $f_2(\boldsymbol{x})$ *равны*, если и только если

$$M_{f_1}^1 = M_{f_2}^1,\ M_{f_1}^0 = M_{f_2}^0,\ M_{f_1}^- = M_{f_2}^-.$$

Будем называть частичные функции $f_1(\boldsymbol{x})$ и $f_2(\boldsymbol{x})$ *взаимно инверсными* ($f_2 = \overline{f_1}$, $f_1 = \overline{f}_2$), если и только если

$$M_{f_1}^1 = M_{f_2}^0,\ M_{f_1}^0 = M_{f_2}^1,\ M_{f_1}^- = M_{f_2}^-.$$

Таким образом, взаимно инверсные частичные функции имеют одну и ту же область неопределенных значений.

Частичная функция f_i *реализуется* частичной (либо полностью определенной) функцией f_j (обозначается $f_i \prec f_j$), если и только если

$$M_{f_i}^1 \subseteq M_{f_j}^1,\ M_{f_i}^0 \subseteq M_{f_j}^0.$$

Функцию f_j называют *доопределением* функции f_i. Для пары полностью определенных булевых функций f_i, f_j отношение реализации является отношением равенства. Пусть требу-

ется проверить отношение реализации $f_i(\boldsymbol{x}) \prec f_j(\boldsymbol{x})$, при этом функции заданы таблицей истинности. В столбцах значений функций $f_i(\boldsymbol{x})$, $f_j(\boldsymbol{x})$ проверим для каждой пары значений этих функций выполнение битовых отношений реализации: $0 \prec 0$; $1 \prec 1$; $- \prec -$; $- \prec 0$; $- \prec 1$.

Если для каждой пары соответствующих значений функций $f_i(\boldsymbol{x})$, $f_j(\boldsymbol{x})$ выполняется одно (любое) из пяти перечисленных битовых отношений, то выполняется и отношение реализации $f_i(\boldsymbol{x}) \prec f_j(\boldsymbol{x})$. Если же функции $f_i(\boldsymbol{x})$, $f_j(\boldsymbol{x})$ таковы, что для каждой пары их значений выполняется $0 \prec 0$ либо $1 \prec 1$, то это означает, что функции $f_i(\boldsymbol{x})$, $f_j(\boldsymbol{x})$ являются *полностью определенными и равными*.

Под *векторной булевой функцией* $\boldsymbol{f}(\boldsymbol{x})$ будем понимать упорядоченную систему частичных булевых функций $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$, значениями векторных функций на элементах $\boldsymbol{x}^*$ булева пространства являются m-компонентные троичные векторы $\boldsymbol{f}(\boldsymbol{x}^*)$. Неупорядоченное множество $\{ f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}) \}$ компонент векторной функции $\boldsymbol{f}(\boldsymbol{x})$ будем обозначать $F(\boldsymbol{x})$ либо F.

Алгебраические разложения частичных функций. *Задача 1.* Заданы частичные булевы функции $\varphi(\boldsymbol{x})$, $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$. Требуется найти, если это возможно, такие доопределения $\varphi^*(\boldsymbol{x})$, $f_1^*(\boldsymbol{x})$, $f_2^*(\boldsymbol{x})$ соответствующих функций (запишем это в виде $\varphi(\boldsymbol{x}) \prec \varphi^*(\boldsymbol{x})$, $f_1(\boldsymbol{x}) \prec f_1^*(\boldsymbol{x})$, $f_2(\boldsymbol{x}) \prec f_2^*(\boldsymbol{x})$), что будет выполняться одно из алгебраических разложений:

$$\varphi^*(\boldsymbol{x}) \prec (f_1^*(\boldsymbol{x}) \vee f_2^*(\boldsymbol{x})), \tag{1}$$

$$\varphi^*(\boldsymbol{x}) \prec (f_1^*(\boldsymbol{x}) \,\&\, f_2^*(\boldsymbol{x})). \tag{2}$$

Естественно, для каждого из разложений (1), (2) требуются свои доопределения функций $\varphi(\boldsymbol{x})$, $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$. Логические операторы $\vee$ (дизъюнкция), & (конъюнкция) в формулах (1), (2) называют также *выходными* функциями соответствующего разложения. В работе [6] логические операции над частичными булевыми функциями (троичными переменными) определены следующим образом:

функция f_1	0 0 0 – – – 1 1 1
функция f_2	0 – 1 0 – 1 0 – 1
отрицание $\overline{f_1}$	1 1 1 – – – 0 0 0
дизъюнкция $f_1 \vee f_2$	0 – 1 – – 1 1 1 1
конъюнкция $f_1 \,\&\, f_2$	0 0 0 0 – – 0 – 1

Для краткости назовем *алгебраическими* разложениями множество дизъюнктивных и конъюнктивных разложений функций заданной системы частичных булевых функций.

Утверждение 1. *Задача нахождения разложения (1) не имеет решения тогда и только тогда, когда выполняется хотя бы одно из условий (3)–(5):*

$$(M_{f_1}^0 \cap M_{f_2}^0) \cap M_{\varphi}^1 \neq \varnothing, \tag{3}$$

$$M_{f_1}^1 \cap M_{\varphi}^0 \neq \varnothing, \tag{4}$$

$$M_{f_2}^1 \cap M_{\varphi}^0 \neq \varnothing. \tag{5}$$

Условие (3) говорит о том, что если найдется хотя бы один набор значений переменных, на котором обе функции $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ принимают значение нуль, а функция $\varphi(\boldsymbol{x})$ принимает значение единица, то через операцию дизъюнкции функций $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ нельзя выразить функцию $\varphi(\boldsymbol{x})$, так как дизъюнкция двух нулей не может быть равна единице. В случае условия (4) от значения функции $f_2(\boldsymbol{x})$ ничего не зависит. Дизъюнкция $f_1(\boldsymbol{x}) \vee f_2(\boldsymbol{x})$ не может

принять значение нуль (значение нуль имеет функция $\varphi(\boldsymbol{x})$), так как $f_1(\boldsymbol{x})$ принимает значение единица, аналогично и для условия (5). Табл. 1 позволяет проверить возможность доопределения функций $\varphi(\boldsymbol{x})$, $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ для проверки реализации $\varphi^*(\boldsymbol{x}) \prec (f_1^*(\boldsymbol{x}) \vee f_2^*(\boldsymbol{x}))$. Рассмотрим третью строку табл. 1, в которой $f_1(\boldsymbol{x})=0$, $f_2(\boldsymbol{x})=0$, $\varphi(\boldsymbol{x})=$«–». Для выполнения дизъюнктивного разложения нужно заменить значение «–» на нуль: $\varphi(\boldsymbol{x})=0$. Аналогично можно доопределять значения «–» и для других строк из табл. 2, не содержащих пометку «Нет решения».

Таблица 1
Доопределение частичных функций для дизъюнктивного разложения
Table 1
Redefinition of partial functions for disjunctive decomposition

Значения исходных булевых функций *Values of the original Boolean functions*			Значения реализующих функций для разложения $\varphi^* \prec (f_1^* \vee f_2^*)$ *Values of implementing functions for decomposition* $\varphi^* \prec (f_1^* \vee f_2^*)$		
f_1	f_2	φ	f_1^*	f_2^*	φ^*
0	0	0	0	0	0
0	0	1	Нет решения		
0	0	–	0	0	**0**
0	1	0	Нет решения		
0	1	1	0	1	1
0	1	–	0	1	**1**
0	–	0	0	**0**	0
0	–	1	0	**1**	1
0	–	–	0 0	**0** **1**	**0** **1**
1	0	0	Нет решения		
1	0	1	1	0	1
1	0	–	1	0	**1**
1	1	0	Нет решения		
1	1	1	1	1	1
1	1	–	1	1	**1**
1	–	0	Нет решения		
1	–	1	1	–	1
1	–	–	1	–	**1**
–	0	0	**0**	0	0
–	0	1	**1**	0	1
–	0	–	**0** **1**	0 0	**0** **1**
–	1	0	Нет решения		
–	1	1	–	1	1
–	1	–	–	1	**1**
–	–	0	**0**	**0**	0
–	–	1	**1** –	– **1**	1 1
–	–	–	–	–	–

Таблица 2
Доопределение частичных функций для конъюнктивного разложения
Table 2
Redefinition of partial functions for conjunctive decomposition

Значения исходных булевых функций *Values of the original Boolean functions*			Значения реализующих функций для разложения $\varphi^* \prec (f_1^* \& f_2^*)$ *Values of implementing functions for decomposition* $\varphi^* \prec (f_1^* \& f_2^*)$		
f_1	f_2	φ	f_1^*	f_2^*	φ^*
0	0	0	0	0	0
0	0	1	Нет решения		
0	0	–	0	0	**0**
0	1	0	0	1	0
0	1	1	Нет решения		
0	1	–	0	1	**0**
0	–	0	0	**0**	0
0	–	1	Нет решения		
0	–	–	0	–	**0**
1	0	0	1	0	0
1	0	1	Нет решения		
1	0	–	1	0	**0**
1	1	0	Нет решения		
1	1	1	1	1	1
1	1	–	1	1	**1**
1	–	0	1	**0**	0
1	–	1	1	**1**	1
1	–	–	1 1	**1** **0**	**1** **0**
–	0	0	**0**	0	0
–	0	1	Нет решения		
–	0	–	–	0	**0**
–	1	0	**0**	1	0
–	1	1	**1**	1	1
–	1	–	**1** **0**	1 1	**1** **0**
–	–	0	**0** –	– **0**	0 0
–	–	1	**1**	**1**	1
–	–	–	–	–	–

Утверждение 2. *Задача нахождения разложения (2) не имеет решения тогда и только тогда, когда верно хотя бы одно из условий:*

$$M^0_{f_1} \bigcap M^1_{\varphi} \neq \varnothing, \quad (6)$$

$$M^0_{f_2} \bigcap M^1_{\varphi} \neq \varnothing, \quad (7)$$

$$(M^1_{f_1} \bigcap M^1_{f_2}) \bigcap M^0_{\varphi} \neq \varnothing. \quad (8)$$

Табл. 2 позволяет проверить возможность доопределения функций $\varphi(\boldsymbol{x})$, $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ для проверки реализации $\varphi^*(\boldsymbol{x}) \prec (f_1^*(\boldsymbol{x}) \,\&\, f_2^*(\boldsymbol{x}))$. Таким образом, решение задачи 1 осуществляется с помощью табл. 1 и 2, позволяющих выполнить проверку и найти доопределенные функции $\varphi^*(\boldsymbol{x})$, $f_1^*(\boldsymbol{x})$, $f_2^*(\boldsymbol{x})$ либо убедиться, что таких функций не существует.

Пусть частичные функции $\varphi^*(\boldsymbol{x})$, $f_1^*(\boldsymbol{x})$, $f_2^*(\boldsymbol{x})$ являются доопределениями соответствующих функций $\varphi(\boldsymbol{x})$, $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$, т. е. выполняются реализации $\varphi(\boldsymbol{x}) \prec \varphi^*(\boldsymbol{x})$, $f_1(\boldsymbol{x}) \prec f_1^*(\boldsymbol{x})$, $f_2(\boldsymbol{x}) \prec f_2^*(\boldsymbol{x})$.

Утверждение 3. *Если для тройки частичных функций* $\varphi(\boldsymbol{x})$, $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ *не существует дизъюнктивное разложение* $\varphi(\boldsymbol{x}) \prec (f_1(\boldsymbol{x}) \vee f_2(\boldsymbol{x}))$, *то и для любых доопределений* $\varphi^*(\boldsymbol{x})$, $f_1^*(\boldsymbol{x})$, $f_2^*(\boldsymbol{x})$ *дизъюнктивные разложения* $\varphi^*(\boldsymbol{x}) \prec (f_1^*(\boldsymbol{x}) \vee f_2^*(\boldsymbol{x}))$ *также не существуют.*

Аналогичное утверждение 4 справедливо и для конъюнктивного разложения.

Утверждение 4. *Если для тройки частичных функций* $\varphi(\boldsymbol{x})$, $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$ *не существует конъюнктивное разложение* $\varphi(\boldsymbol{x}) \prec (f_1(\boldsymbol{x}) \,\&\, f_2(\boldsymbol{x}))$, *то и для любых доопределений* $\varphi^*(\boldsymbol{x})$, $f_1^*(\boldsymbol{x})$, $f_2^*(\boldsymbol{x})$ *конъюнктивные разложения* $\varphi^*(\boldsymbol{x}) \prec (f_1^*(\boldsymbol{x}) \,\&\, f_2^*(\boldsymbol{x}))$ *также не существуют.*

Нахождение раздельных алгебраических разложений. Рассмотрим систему $F=\{f_1,\ldots,f_k\}$ частичных булевых функций. Чтобы найти все варианты алгебраических разложений, требуется перебор всех различных троек функций из системы F. Пусть k – число функций $f_1,\ldots,f_k$. Тогда требуется рассмотреть $3\times C_k^3$ (C_k^3 – число сочетаний из k функций по три, $k \geq 3$) вариантов проверки дизъюнктивных разложений: число всех различных неупорядоченных троек функций f_p, f_i, f_j равно C_k^3, а для каждой тройки $\{f_p, f_i, f_j\}$ надо проверять три дизъюнктивных разложения $f_p = f_i \vee f_j$; $f_i = f_p \vee f_j$; $f_j = f_i \vee f_p$. Аналогично для каждой из C_k^3 троек функций надо перебирать три варианта при построении конъюнктивных разложений. Общий перебор составит $6C_k^3$ вариантов. Существование отдельных разложений можно проверять, используя табл. 1 и 2, в которых приводятся требуемые доопределения частичных функций для выполнения дизъюнктивных (табл. 1) и конъюнктивных (табл. 2) разложений.

Пусть оператор $\odot$ обозначает любую из рассматриваемых алгебраических операций – конъюнкцию либо дизъюнкцию (рис. 1).

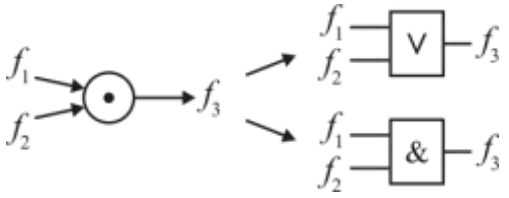


Рис. 1. Интерпретация оператора $\odot$

Fig. 1. Interpretation of the operator $\odot$

Обозначим через E множество найденных разложений $E = \{E_1, \ldots, E_q\}$, которые назовем *элементарными*. В алгебраическом разложении $f_p^* \prec f_i^* \odot f_j^*$ участвует тройка T функций. Функцию f_p^* элементарного разложения $f_p^* \prec f_i^* \odot f_j^*$ назовем *представимой*. Если для функции f_p системы F не найдено ни одного алгебраического разложения из функций системы F, то такую функцию назовем *непредставимой*. Однако непредставимые функции могут участвовать в разложениях представимых функций. Каждое элементарное разложение $f_p^* \prec f_i^* \odot f_j^*$ характеризуется парой неотрицательных целочисленных весов w_{f_i, f_i^*}, w_{f_j, f_j^*} для доопределенных функций f_i^*, f_j^*, которые назовем *базисными* функциями этого разложения. Вес w_{f_i, f_i^*} задает число наборов, на которых функция f_i доопределена до функции f_i^*, вес w_{f_j, f_j^*} – число наборов, на которых функция f_j доопределена до функции f_j^*. Если вес равен нулю, то для существования разложения данную функцию не потребовалось доопределять. Если функция f_p является непредставимой и не участвует ни в одном из разложений ни одной из представимых функций, то такие функции f_p удаляются из рассмотрения, так как предлагаемые далее алгоритмы для этих функций не предназначены. Представимые функции могут быть заменены дизъюнкциями либо конъюнкциями других функций. Дизъюнктивное разложение при схемной реализации соответствует логическому элементу ИЛИ (дизъюнктору), конъюнктивное – элементу И (конъюнктору).

Пару элементарных алгебраических разложений E_1, E_2 назовем раздельной (либо будем говорить, что разложения E_1, E_1 являются *раздельными*), если тройки T_1, T_2 участвующих в этих разложениях функций не пересекаются: $T_1 \bigcap T_2 = \varnothing$. Очевидно, что в каждом раздельном разложении участвует своя тройка функций. Рассмотрим множество R^{sep} попарно *раздельных* разложений, т. е. таких, для которых попарные пересечения троек функций для каждой пары различных разложений из R^{sep} являются пустыми.

Пример 1. Пусть f_1, f_2, f_3, f_4, f_5, f_6, f_7 являются кофакторами разложения Шеннона частичных функций r_1, r_2, r_3, r_4, r_5 по переменной x_5 (рис. 2), т. е. $r_1 = \overline{x}_5 f_1 \vee x_5 f_2$, $r_2 = \overline{x}_5 f_2 \vee x_5 f_3$, $r_3 = \overline{x}_5 f_3 \vee x_5 f_5$, $r_4 = \overline{x}_5 f_4 \vee x_5 f_3$, $r_5 = \overline{x}_5 f_6 \vee x_5 f_7$. Требуется проверить, существуют ли элементарные разложения

$$f_3^* \prec (f_1^* \vee f_2^*),\ f_7^* \prec (f_5^* \vee f_6^*)$$

для функций f_1, f_2, f_3, f_4, f_5, f_6, f_7, заданных в табл. 3, и найти соответствующие доопределения f_1^*, f_2^*, f_3^*, f_5^*, f_6^*, f_7^*.

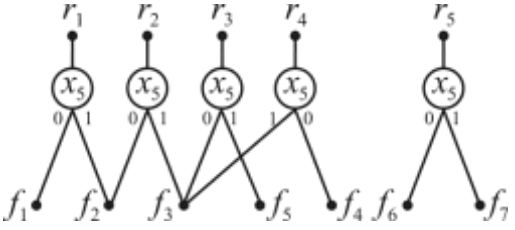


Рис. 2. Бинарная диаграмма решений – разложение по переменной x_5

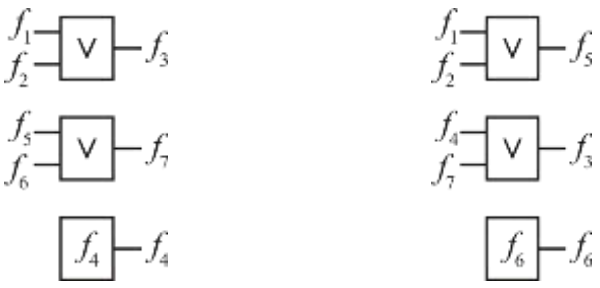
Fig. 2. Binary Decision Diagram – decomposition by variable x_5

Таблица 3
Система частичных булевых функций
Table 3
A system of partial Boolean functions

$x_1\ x_2\ x_3\ x_4$	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0 0 0 0	–	–	–	–	–	–	–
0 0 0 1	0	–	1	–	0	1	–
0 0 1 0	1	0	1	1	–	0	–
0 0 1 1	–	–	–	–	–	–	–
0 1 0 0	0	0	–	1	0	–	1
0 1 0 1	–	–	–	–	–	–	–
0 1 1 0	–	1	–	1	1	0	–
0 1 1 1	–	–	–	–	–	–	–
1 0 0 0	–	0	1	0	0	–	1
1 0 0 1	–	–	–	–	0	–	–
1 0 1 0	0	0	0	0	–	–	0
1 0 1 1	–	–	–	–	1	1	–
1 1 0 0	–	–	–	–	–	–	–
1 1 0 1	1	1	–	1	–	–	–
1 1 1 0	–	–	–	–	–	1	–
1 1 1 1	–	–	–	–	–	0	1

В табл. 4 даны доопределения (показаны в скобках) функций, найденные с помощью табл. 1. Таким образом, из существования доопределений f_1^*, f_2^*, f_3^*, f_5^*, f_6^*, f_7^* функций следует, что элементарные разложения существуют. На наборе 0010 функции f_5^* и f_7^* имеют значение нуль, хотя согласно табл. 1 они могут быть доопределены и до единицы. Аналогичная ситуация на наборе 1001 для функций f_6^*, f_7^*, которые также доопределены до нуля. Заметим, что данные функции являются кофакторами в BDD по переменной x_5 (см. рис. 2) и будут подвергаться дальнейшим разложениям Шеннона. Как показано в работе [1], нулевые доопределения частичных булевых функций часто (но не всегда) являются целесообразными, так как уменьшают число кофакторов при дальнейшей минимизации BDD-представлений булевых функций.

Поэтому две представимые функции f_3, f_7 не потребуют дальнейшей оптимизации, так как могут быть реализованы с помощью дизъюнкторов. Функция f_4 не участвует в разложениях, поэтому не была доопределена. Сумма весов доопределений базисных функций равна девяти. Логическая схема, соответствующая найденным разложениям, показана на рис. 3, *а*.



a) *b)*

Рис. 3. Логические схемы: *а)* для первого множества раздельных разложений;
б) для второго множества раздельных разложений

Fig. 3. Logic circuits: a) for the first set of separate decompositions;
b) for the second set of separate decompositions

Таблица 4
Доопределенные функции из табл. 3
Table 4
Predefined functions from Table 3

$x_1\ x_2\ x_3\ x_4$	f_1^*	f_2^*	f_3^*	f_4	f_5^*	f_6^*	f_7^*
0 0 0 0	–	–	–	–	–	–	–
0 0 0 1	0	–(1)	1	–	0	1	–(1)
0 0 1 0	1	0	1	1	–(0)	0	–(0)
0 0 1 1	–	–	–	–	–	–	–
0 1 0 0	0	0	–(0)	1	0	–(1)	1
0 1 0 1	–	–	–	–	–	–	–
0 1 1 0	–	1	–(1)	1	1	0	–(1)
0 1 1 1	–	–	–	–	–	–	–
1 0 0 0	–(1)	0	1	0	0	–(1)	1
1 0 0 1	–	–	–	–	0	–(0)	–(0)
1 0 1 0	0	0	0	0	–(0)	–(0)	0
1 0 1 1	–	–	–	–	1	1	–(1)
1 1 0 0	–	–	–	–	–	–	–
1 1 0 1	1	1	–(1)	1	–	–	–
1 1 1 0	–	–	–	–	–	1	–(1)
1 1 1 1	–	–	–	–	–(1)	0	1
Вес w_{f_i}	1	1	3	0	3	4	6

Множество $R_{\max}^{sep}$ раздельных разложений для заданной системы F частичных функций назовем *максимальным* (по мощности), если оно содержит наибольшее число раздельных разложений, т. е. если в множестве $R_{\max}^{sep}$ имеется наибольшее число различных троек функций из системы F, причем одна из функций тройки представима в виде дизъюнкции либо конъюнкции двух оставшихся функций данной тройки.

Сформулируем задачу 2 нахождения максимального множества $R_{\max}^{sep}$ раздельных разложений для заданной системы $F=\{f_1,\ldots,f_k\}$ частичных булевых функций. Каждая представимая функция из множества $R_{\max}^{sep}$ при этом должна задаваться в алгебраическом (дизъюнктивном либо конъюнктивном) разложении своей парой базисных функций, т. е. каждая базисная функция может входить в разложение только одной представимой функции из $R_{\max}^{sep}$. В таком случае в схеме не будет соединений между логическими элементами И, ИЛИ.

Задача 2. Для заданной системы $F=\{f_1,\ldots,f_k\}$ частичных функций требуется найти такое максимальное по мощности множество $R_{\max}^{sep}$ раздельных алгебраических разложений, чтобы была минимальной сумма весов базисных функций.

Зададим найденные элементарные разложения булевой матрицей B, столбцы которой соответствуют функциям $f_1,\ldots,f_k$ системы F, строки – разложениям, а единичные значения элементов строки отмечают функции, входящие в соответствующее элементарное разложение. Если существуют два разложения вида $f_p^* \prec f_i^* \vee f_j^*$, $f_p^* \prec f_i^* \,\&\, f_j^*$, то в матрице B оставляется строка с меньшим суммарным весом $w_{f_i,f_i^*} + w_{f_j,f_j^*}$. Если для этих разложений суммы $w_{f_i,f_i^*} + w_{f_j,f_j^*}$ весов одинаковы, то удаляется одна (любая) из строк.

Решение задачи 2 сводится к нахождению такого строчного покрытия столбцов матрицы *B*, чтобы максимальное число этих столбцов было покрыто строками, не имеющими попарных пересечений, и чтобы был минимальным суммарный вес строк найденного покрытия. Напомним, что весом строки матрицы *B* (т. е. соответствующего ей дизъюнктивного либо конъюнктивного разложения) является суммарное число доопределяемых значений двух базисных функций, связанных дизъюнкцией либо конъюнкцией. Методы нахождения взвешенных строчных покрытий булевых матриц рассматривались в работе [7] для случая, когда не было ограничения на вид покрытия, а именно когда строки покрытия попарно не пересекались.

Эвристический алгоритм решения задачи 2. Алгоритм состоит в итеративном повторении шагов 1 и 2, пока матрица *B* не станет пустой (не будет содержать строк):

Шаг 1. Найти строку (тройку функций) матрицы *B* с минимальным весом и зачислить ее в искомое покрытие.

Шаг 2. Удалить все те строки матрицы *B*, в которых содержится хотя бы одна из функций найденной на шаге 1 тройки функций.

Рассмотрим решение задачи 2 на примере системы функций, заданных в табл. 3. Не гарантируя полноту проверки всех $6C_7^3 = 210$ возможных вариантов алгебраических разложений, зададим найденные 30 вариантов разложений и их веса в виде табл. 5. Например, функция f_5 может быть выражена в виде разложения $f_5^* \prec (f_1^* \vee f_2^*)$ с суммой весов $w_{f_1, f_1^*} + w_{f_2, f_2^*} = 3$.

Таблица 5
Матрица *B* покрытия функций элементарными разложениями (нули заменены точками)
Table 5
Matrix B of function coverage by elementary decompositions (zeros replaced by dots)

f_1	f_2	f_3	f_4	f_5	f_6	f_7	Разложение *Decomposition*	$w_{f_i, f_i^*} + w_{f_j, f_j^*}$
Матрица *B* *Matrix B*								
1	.	1	.	1	.	.	$E_1 = \{ f_1^* \prec (f_3^* \& f_5^*) \}$	8
1	.	1	.	.	.	1	$E_2 = \{ f_1^* \prec (f_3^* \& f_7^*) \}$	6
1	.	.	1	1	.	.	$E_3 = \{ f_1^* \prec (f_4^* \& f_5^*) \}$	6
1	.	.	.	1	.	1	$E_4 = \{ f_1^* \prec (f_5^* \& f_7^*) \}$	9
1	1	.	.	1	.	.	$E_5 = \{ f_2^* \prec (f_1^* \& f_5^*) \}$	7
1	1	1	.	.	.	.	$E_6 = \{ f_3^* \prec (f_1^* \vee f_2^*) \}$	2
1	.	1	.	.	1	.	$E_7 = \{ f_3^* \prec (f_1^* \vee f_6^*) \}$	4
.	.	1	1	.	1	.	$E_8 = \{ f_3^* \prec (f_4^* \vee f_6^*) \}$	4
.	.	1	1	.	.	1	$E_9 = \{ f_3^* \prec (f_4^* \vee f_7^*) \}$	1
.	.	1	.	1	1	.	$E_{10} = \{ f_3^* \prec (f_5^* \vee f_6^*) \}$	7
.	.	1	.	1	.	1	$E_{11} = \{ f_3^* \prec (f_5^* \vee f_7^*) \}$	3
1	.	.	1	.	1	.	$E_{12} = \{ f_4^* \prec (f_1^* \vee f_6^*) \}$	6
.	.	.	1	1	1	.	$E_{13} = \{ f_4^* \prec (f_5^* \vee f_6^*) \}$	7
1	1	.	.	1	.	.	$E_{14} = \{ f_5^* \prec (f_1^* \vee f_2^*) \}$	3
1	1	.	.	1	.	.	$E_{15} = \{ f_5^* \prec (f_1^* \& f_3^*) \}$	9
1	.	.	.	1	.	1	$E_{16} = \{ f_5^* \prec (f_1^* \& f_7^*) \}$	11
.	1	.	1	1	.	.	$E_{17} = \{ f_5^* \prec (f_2^* \& f_4^*) \}$	6

Окончание табл. 5
End of table 5

f_1	f_2	f_3	f_4	f_5	f_6	f_7	Разложение *Decomposition*	$w_{f_i,f_i^*}+w_{f_j,f_j^*}$
Матрица *B* *Matrix B*								
.	1	.	.	1	.	1	$E_{18}=\{f_5^* \prec (f_2^* \& f_7^*)\}$	7
.	.	1	1	1	.	.	$E_{19}=\{f_5^* \prec (f_3^* \& f_4^*)\}$	8
.	1	.	.	.	1	1	$E_{20}=\{f_6^* \prec (f_2^* \& f_7^*)\}$	9
1	.	1	.	.	.	1	$E_{21}=\{f_7^* \prec (f_1^* \vee f_3^*)\}$	2
1	.	.	1	.	.	1	$E_{22}=\{f_7^* \prec (f_1^* \vee f_4^*)\}$	4
1	.	.	.	.	1	1	$E_{23}=\{f_7^* \prec (f_1^* \vee f_6^*)\}$	5
.	1	1	.	.	.	1	$E_{24}=\{f_7^* \prec (f_2^* \vee f_3^*)\}$	2
.	1	.	.	.	1	1	$E_{25}=\{f_7^* \prec (f_2^* \vee f_6^*)\}$	4
.	.	1	1	.	.	1	$E_{26}=\{f_7^* \prec (f_3^* \vee f_4^*)\}$	1
.	.	1	.	1	.	1	$E_{27}=\{f_7^* \prec (f_3^* \vee f_5^*)\}$	4
.	.	1	.	.	1	1	$E_{28}=\{f_7^* \prec (f_3^* \vee f_6^*)\}$	4
.	.	.	1	.	1	1	$E_{29}=\{f_7^* \prec (f_4^* \vee f_6^*)\}$	3
.	.	.	.	1	1	1	$E_{30}=\{f_7^* \prec (f_5^* \vee f_6^*)\}$	7

На шаге 1 занесем в покрытие строку с разложением E_9, имеющую вес единица. На шаге 2 получим сокращенную матрицу *B*, состоящую из двух строк (табл. 6).

Таблица 6
Матрица *B* в начале итерации 2
Table 6
Matrix B at the beginning of iteration 2

f_1	f_2	f_3	f_4	f_5	f_6	f_7	Разложение *Decomposition*	$w_{f_i,f_i^*}+w_{f_j,f_j^*}$
Матрица *B* *Matrix B*								
1	1	.	.	1	.	.	$E_5=\{f_2^* \prec (f_1^* \& f_5^*)\}$	7
1	1	.	.	1	.	.	$E_{14}=\{f_5^* \prec (f_1^* \vee f_2^*)\}$	3

На итерации 2 в матрице *B* (табл. 6) выбираем строку с разложением E_{14} и весом три, сокращаем матрицу – она становится пустой. Результат эвристического алгоритма решения задачи 2: $R_{\max}^{sep}=\{E_9,\ E_{14}\}$, где $E_9=\{f_3^* \prec (f_4^* \vee f_7^*)\}$, $E_{14}=\{f_5^* \prec (f_1^* \vee f_2^*)\}$ с суммарным весом четыре. Столбец f_6 матрицы *B* не покрыт, поэтому функция f_6 не вошла ни в одно из разложений найденного множества $R_{\max}^{sep}$. Очевидно, что тройки $T_9=\{f_3^*,\ f_4^*,\ f_7^*\}$, $T_{14}=\{f_1^*,\ f_2^*,\ f_5^*\}$ функций, из которых состоят раздельные разложения E_9, E_{14}, являются различными (не пересекаются). Назовем раздельные разложения E_9, E_{14} *вторым множеством раздельных разложений*, соответствующая логическая схема показана на рис. 3, *b*. Базисные функции второго множества раздельных разложений имеют вид

x_1	x_2	x_3	x_4	f_1^*	f_2^*	f_4^*	f_6	f_7^*
0	0	0	0	–	–	–	–	–
0	0	0	1	0	0	–	1	1
0	0	1	0	1	0	1	0	–
0	0	1	1	–	–	–	–	–
0	1	0	0	0	0	1	–	1
0	1	0	1	–	–	–	–	–
0	1	1	0	-1	1	0	–	
0	1	1	1	–	–	–	–	–
1	0	0	0	0	0	0	–	1
1	0	0	1	0	0	–	–	–
1	0	1	0	0	0	0	–	0
1	0	1	1	–	1	–	1	–
1	1	0	0	–	–	–	–	–
1	1	0	1	1	1	1	–	–
1	1	1	0	–	–	–	1	–
1	1	1	1	–	–	–	0	1

Суммарный вес раздельных разложений из множества $R_{\max}^{sep} = \{E_9, E_{14}\}$ равен четырем. Заметим, что ранее найденные элементарные разложения $f_3^* \prec (f_1^* \vee f_2^*)$, $f_7^* \prec (f_5^* \vee f_6^*)$ также являются раздельными (см. рис. 3, *а*), однако суммарный вес базисных функций f_1^*, f_2^*, f_5^*, f_6^* равен девяти (см. табл. 4).

Виды совместных разложений. В отличие от раздельных разложений рассматриваемые далее *совместные* разложения имеют *общие* (совместно используемые) функции для элементарных разложений.

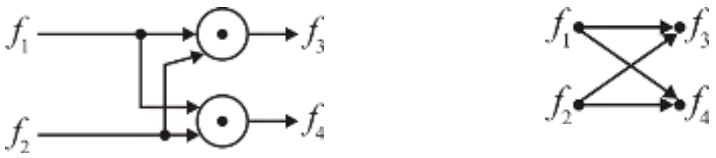
Совокупности взаимосвязанных алгебраических разложений совместного разложения соответствует логическая схема в базисе элементов И, ИЛИ, при этом представимым функциям соответствуют выходные и промежуточные переменные схемы. Функции, которые соответствуют входным полюсам схемы, назовем *базисными*. Если рассматривать логические уравнения, соответствующие разложениям, то вместо комбинационной схемы лучше использовать ориентированный граф (орграф) G зависимости разлагаемых функций от базисных (и уже разложенных), вершины которого соответствуют функциям, а дуги – связям между функциями: две заходящие дуги соответствуют двум функциям, которые составляют разложение той функции, в которую дуги заходят.

Утверждение 5. *Орграф G для совместного разложения должен быть бесконтурным (не должен содержать контуров).*

Далее будут рассматриваться только те виды совместных разложений, которые показаны на рис. 4–7. Очевидно, что они не исчерпывают все возможные совместные разложения для систем (множеств) функций мощности четыре–семь, тем более для систем функций большей мощности. На рис. 4, *b* и 5, *b* показаны графы G. Очевидно, что они не содержат контуров, как и графы для других видов (рис. 6 и 7) совместных разложений.

Совместное разложение, показанное на рис. 4, *а*, имеет две общие функции для двух элементарных разложений. На рис. 5 изображена структура совместного разложения с одной общей функцией f_4, позволяющая получить при конкретизации оператора $\odot$ четыре совместных разложения:

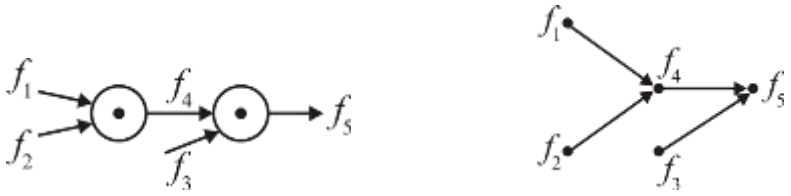
1. $f_4 \prec (f_1 \vee f_2)$; $f_5 \prec (f_3 \vee f_4)$;
2. $f_4 \prec (f_1 \& f_2)$; $f_5 \prec (f_3 \vee f_4)$;
3. $f_4 \prec (f_1 \vee f_2)$; $f_5 \prec (f_3 \& f_4)$;
4. $f_4 \prec (f_1 \& f_2)$; $f_5 \prec (f_3 \& f_4)$.



a) *b)*

Рис. 4. Совместное разложение: *a)* реализация двумя базисными функциями f_1, f_2 двух представимых функций f_3, f_4; *b)* граф *G* зависимости функций

Fig. 4. Joint decomposition: a) implementation by two basic functions f_1, f_2 two representable functions f_3, f_4; b) graph G of function dependence



a) *b)*

Рис. 5. Совместное разложение: *a)* реализация тремя базисными функциями f_1, f_2, f_3 двух представимых функций f_4, f_5; *b)* граф *G* зависимости функций

Fig. 5. Joint decomposition: a) implementation by three basic functions f_1, f_2, f_3 of two representable functions f_4, f_5; b) graph G of function dependence

Рассмотрим совместное разложение на рис. 6, *b*. Для разложений $f_4 = f_1 \odot f_2$ $f_6 = f_1 \odot f_5$ имеется одна общая функция f_1. Функция f_2 является общей для разложений $f_5 = f_2 \odot f_3$, $f_4 = f_1 \odot f_2$. Аналогичный анализ может быть проведен для других совместных разложений, показанных на рис. 4–7.

Рис. 6. Совместные разложения для реализации трех представимых функций f_4, f_5, f_6

Fig. 6. Joint decompositions to implement three representable functions f_4, f_5, f_6

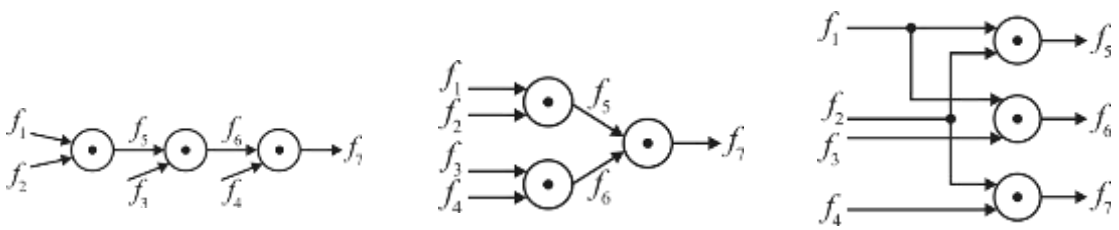


Рис. 7. Совместные разложения для реализации трех представимых функций f_5, f_6, f_7

Fig. 7. Joint decompositions to implement three representable functions f_5, f_6, f_7

Проверка существования совместного разложения. *Задача 3.* Требуется проверить, существует ли совместное разложение заданного вида, и если существует, то найти доопределения частичных функций, входящих в совместное разложение.

Очевидно, что каждое из отдельных разложений, входящих в совместное разложение, должно быть элементарным, т. е. для троек функций разложений должны существовать соответствующие доопределения – это необходимые условия существования совместного разложения. Покажем на простом примере, что раздельные разложения могут существовать, а совместное – нет. В табл. 7 приведены булевы функции, составляющие два разложения. Рассмотрим *совместное* разложение $f_4 = f_1 \& f_2$, $f_5 = f_2 \& f_3$, такое, которое соответствует разложению, показанному на рис. 6, *а*. *Раздельное* разложение $f_4 = f_1 \& f_2$ существует, если заменить неопределенное значение «–» функции f_2 на значение единица; *раздельное* разложение $f_5 = f_2 \& f_3$ также существует, если заменить неопределенное значение «–» функции f_2 на значение нуль. Очевидно, что общее согласованное доопределение не существует из-за того, что требуется противоречивое доопределение функции f_2. Следовательно, совместное разложение $f_4 = f_1 \& f_2$, $f_5 = f_2 \& f_3$ не существует, хотя существуют (по отдельности) требуемые доопределения для указанных раздельных разложений.

Таблица 7
Раздельные разложения
Table 7
Separate decompositions

$x_1\ x_2$	$f_4 = f_1 \& f_2$			$f_5 = f_2 \& f_3$		
	f_1	f_2	f_4	f_2	f_3	f_5
0 0	1	1	1	1	0	0
0 1	1	0	0	0	0	0
1 0	0	1	0	1	1	1
1 1	1	–	1	–	1	0

Утверждение 6. *Функции, участвующие в совместном разложении, должны иметь хотя бы одно общее согласованное доопределение, которое удовлетворяет каждому элементарному алгебраическому разложению, входящему в совместное разложение.*

Решение задачи 3 сводится к составлению и решению логического уравнения, задающего условия согласованного доопределения всех частичных функций, участвующих в совместном разложении. Существование хотя бы одного решения логического уравнения является достаточным условием существования совместного разложения.

Первое совместное разложение. Проверим, существуют ли для трех заданных формул $f_3 = f_1 \vee f_2$, $f_5 = f_2 \& f_4$, $f_7 = f_1 \vee f_6$ такие доопределения (помечены символом *) частичных функций из табл. 3, чтобы выполнялось совместное разложение вида

$$P_1 = \{ f_3^* \prec (f_1^* \vee f_2^*);\ f_5^* \prec (f_2^* \ \& \ f_4^*);\ f_7^* \prec (f_1^* \vee f_6^*) \}. \qquad (9)$$

Назовем совместное разложение (9) *первым* совместным разложением. Такого вида совместные разложения изображены на рис. 7 (справа). Заметим, что элементарные разложения $E_6 = \{ f_3^* \prec (f_1^* \vee f_2^*) \}$, $E_{17} = \{ f_5^* \prec (f_2^* \& f_4^*) \}$, $E_{23} = \{ f_7^* \prec (f_1^* \vee f_6^*) \}$ существуют (см. табл. 5). Задача заключается в нахождении таких функций f_1^*, f_2^*, f_3^*, f_4^*, f_5^*, f_6^*, f_7^*, чтобы существовало совместное разложение (9).

Будем рассматривать сокращенную табл. 3, удалив строки, для которых все функции системы имеют неопределенное значение «–». Булевы переменные $t_1, \ldots, t_{40}$ для проверки условий существования первого совместного разложения введены для тех значений частичных булевых

функций, которые, возможно, будут доопределены (табл. 8). Заметим, что для тех наборов, на которых значения всех функций не определены, булевы переменные не вводятся – эти строки были удалены из рассмотрения.

Таблица 8
Введение логических переменных для неопределенных значений функций
Table 8
Introduction of logical variables for undefined function values

$x_1\ x_2\ x_3\ x_4$	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0 0 0 1	0	$-(t_1)$	1	$-(t_2)$	0	1	$-(t_3)$
0 0 1 0	1	0	1	1	$-(t_4)$	0	$-(t_5)$
0 1 0 0	0	0	$-(t_6)$	1	0	$-(t_7)$	1
0 1 1 0	$-(t_8)$	1	$-(t_9)$	1	1	0	$-(t_{10})$
1 0 0 0	$-(t_{11})$	0	1	0	0	$-(t_{12})$	1
1 0 0 1	$-(t_{13})$	$-(t_{14})$	$-(t_{15})$	$-(t_{16})$	0	$-(t_{17})$	$-(t_{18})$
1 0 1 0	0	0	0	0	$-(t_{19})$	$-(t_{20})$	0
1 0 1 1	$-(t_{21})$	$-(t_{22})$	$-(t_{23})$	$-(t_{24})$	1	1	$-(t_{25})$
1 1 0 1	1	1	$-(t_{26})$	1	$-(t_{27})$	$-(t_{28})$	$-(t_{29})$
1 1 1 0	$-(t_{30})$	$-(t_{31})$	$-(t_{32})$	$-(t_{33})$	$-(t_{34})$	1	$-(t_{35})$
1 1 1 1	$-(t_{36})$	$-(t_{37})$	$-(t_{38})$	$-(t_{39})$	$-(t_{40})$	0	1

Будем рассматривать строки табл. 8 по отдельности и для каждого разложения записывать требуемые условия доопределения значений функций данной строки, используя табл. 1 для дизъюнктивных разложений и табл. 2 для конъюнктивных разложений. По условиям доопределения будем составлять логические уравнения. Имеющиеся значения и доопределенные значения функций из рассматриваемых строк должны удовлетворять элементарным разложениям, поэтому требуется, чтобы составленное для этих строк логическое уравнение имело хотя бы одно решение. Табл. 9 задает первую итерацию составления логических уравнений по строкам табл. 8.

Логические условия, составленные по строкам, должны выполняться одновременно, поэтому результирующее логическое уравнение после выполнения итерации 1 имеет вид конъюнкции левых частей всех уравнений из табл. 9:

$$t_1(\bar{t}_1 \vee \bar{t}_2)\, t_3\ \bar{t}_4\ t_5\ \bar{t}_6\ t_7\ t_9(\bar{t}_8\bar{t}_{10} \vee t_8 t_{10})\, t_{11}(t_{11} \vee t_{12})(\bar{t}_{14} \vee \bar{t}_{16})\& \\ \& \bar{t}_{19}\ \bar{t}_{20}\ t_{22}\ t_{24}\ t_{25}\ t_{26}\ t_{27}\ t_{29}\ t_{35}\ t_{36} = 1. \qquad (10)$$

Переменные t_{13}, t_{15} t_{17}, t_{18}, t_{21}, t_{23}, t_{28}, t_{30}, t_{31}, t_{32}, t_{33}, t_{34}, t_{37}, t_{38}, t_{39}, t_{40} не участвуют в записи уравнения, составленного на первой итерации. Если у уравнения (10) нет решения, то дальнейший поиск доопределений функций не имеет смысла. Однако уравнение (10) имеет несколько корней (решений), одно из них приведено ниже:

$$t_1 = 1,\ t_2 = 0,\ t_3 = 1,\ t_4 = 0,\ t_5 = 1,\ t_6 = 0,\ t_7 = 1,\ t_8 = 0,\ t_9 = 1,\ t_{10} = 0,\ t_{11} = 1,\ t_{12} = 1,\ t_{14} = 0,\ t_{16} = 0,\ t_{19} = 0, \\ t_{20} = 0,\ t_{22} = 1,\ t_{24} = 1,\ t_{25} = 1,\ t_{26} = 1,\ t_{27} = 1,\ t_{29} = 1,\ t_{35} = 1,\ t_{36} = 1.$$

Таблица 9
Итерация 1 – проверка существования элементарных разложений
Table 9
Iteration 1 – checking the existence of elementary decompositions

Строка таблицы истинности *Row of the truth table*		Разложение $f_3^* \prec (f_1^* \vee f_2^*)$ *Decomposition* $f_3^* \prec (f_1^* \vee f_2^*)$			Разложение $f_5^* \prec (f_2^* \,\&\, f_4^*)$ *Decomposition* $f_5^* \prec (f_2^* \,\&\, f_4^*)$			Разложение $f_7^* \prec (f_1^* \vee f_6^*)$ *Decomposition* $f_7^* \prec (f_1^* \vee f_6^*)$		
		f_1	f_2	f_3	f_2	f_4	f_5	f_1	f_6	f_7
0001	Значения функций	0	t_1	1	t_1	t_2	0	0	1	t_3
	Условия доопределения	$t_1=1$			$(t_1=0)$ или $(t_2=0)$			$t_3=1$		
	Уравнение	$t_1(\bar{t}_1 \vee \bar{t}_2)t_3=1$								
0010	Значения функций	1	0	1	0	1	t_4	1	0	t_5
	Условия доопределения	нет			$t_4=0$			$t_5=1$		
	Уравнение	$\bar{t}_4\, t_5=1$								
0100	Значения функций	0	0	t_6	0	1	0	0	t_7	1
	Условия доопределения	$t_6=0$			нет			$t_7=1$		
	Уравнение	$\bar{t}_6\, t_7=1$								
0110	Значения функций	t_8	1	t_9	1	1	1	t_8	0	t_{10}
	Условия доопределения	$t_9=1$			нет			$((t_8=0)$ и $(t_{10}=0)$ или $((t_8=1)$ и $(t_{10}=1)$		
	Уравнение	$t_9(\bar{t}_8\bar{t}_{10} \vee t_8t_{10})=1$								
1000	Значения функций	t_{11}	0	1	0	0	0	t_{11}	t_{12}	1
	Условия доопределения	$t_{11}=1$			нет			$(t_{11}=1)$ или $(t_{12}=1)$		
	Уравнение	$t_{11}(t_{11} \vee t_{12})=1$								
1001	Значения функций	t_{13}	t_{14}	t_{15}	t_{14}	t_{16}	0	t_{13}	t_{17}	t_{18}
	Условия доопределения	Нет на итерации 1. Появятся на итерации 2, так как $t_{14}=0$			$(t_{14}=0)$ или $(t_{16}=0)$			нет		
	Уравнение	$\bar{t}_{14} \vee \bar{t}_{16}=1$								
1010	Значения функций	0	0	0	0	0	t_{19}	0	t_{20}	0
	Условия доопределения	нет			$t_{19}=0$			$t_{20}=0$		
	Уравнение	$\bar{t}_{19}\, \bar{t}_{20}=1$								

Окончание табл. 9
End of table 9

Строка таблицы истинности *Row of the truth table*		Разложение $f_3^* \prec (f_1^* \vee f_2^*)$ *Decomposition* $f_3^* \prec (f_1^* \vee f_2^*)$			Разложение $f_5^* \prec (f_2^* \,\&\, f_4^*)$ *Decomposition* $f_5^* \prec (f_2^* \,\&\, f_4^*)$			Разложение $f_7^* \prec (f_1^* \vee f_6^*)$ *Decomposition* $f_7^* \prec (f_1^* \vee f_6^*)$		
		f_1	f_2	f_3	f_2	f_4	f_5	f_1	f_6	f_7
1011	Значения функций	t_{21}	t_{22}	t_{23}	t_{22}	t_{24}	1	t_{21}	1	t_{25}
	Условия доопределения	Нет на итерации 1. Появятся на итерации 2, так как $t_{22}=1$			$(t_{22}=1)$ и $(t_{24}=1)$			$t_{25}=1$		
	Уравнение	$t_{22}\, t_{24}\, t_{25}=1$								
1101	Значения функций	1	1	t_{26}	1	1	t_{27}	1	t_{28}	t_{29}
	Условия доопределения	$t_{26}=1$			$t_{27}=1$			$t_{29}=1$		
	Уравнение	$t_{26}\, t_{27}\, t_{29}=1$								
1110	Значения функций	t_{30}	t_{31}	t_{32}	t_{31}	t_{33}	t_{34}	t_{30}	1	t_{35}
	Условия доопределения	нет			нет			$t_{35}=1$		
	Уравнение	$t_{35}=1$								
1111	Значения функций	t_{36}	t_{37}	t_{38}	t_{37}	t_{39}	t_{40}	t_{36}	0	1
	Условия доопределения	Нет на итерации 1. Появятся на итерации 2, так как $t_{36}=1$			нет			$t_{36}=1$		
	Уравнение	$t_{36}=1$								

Таблица 10
Итерация 2. Шаг 1
Table 10
Iteration 2. Step 1

Строка таблицы истинности *Row of the truth table*		Разложение $f_3^* \prec (f_1^* \vee f_2^*)$ *Decomposition* $f_3^* \prec (f_1^* \vee f_2^*)$			Разложение $f_5^* \prec (f_2^* \,\&\, f_4^*)$ *Decomposition* $f_5^* \prec (f_2^* \,\&\, f_4^*)$			Разложение $f_7^* \prec (f_1^* \vee f_6^*)$ *Decomposition* $f_7^* \prec (f_1^* \vee f_6^*)$		
		f_1	f_2	f_3	f_2	f_4	f_5	f_1	f_6	f_7
1001	Значения функций	t_{13}	$t_{14}=0$	t_{15}						
	Условия доопределения	$((t_{13}=0)$ и $(t_{14}=0)$ и $(t_{15}=0))$ или $((t_{13}=1)$ и $(t_{14}=0)$ и $(t_{15}=1))$			нет			нет		
	Уравнение	$\bar{t}_{13}\bar{t}_{14}\bar{t}_{15} \vee t_{13}\bar{t}_{14}t_{15}=1$								

На итерации 2 на каждом из шагов 1–3 (табл. 10–12) рассматриваются те строки, в которых нужны доопределения функций, входящих в первое разложение $f_3^* \prec (f_1^* \vee f_2^*)$. Для функций этого разложения составляются дополнительные условия доопределения, которые вызваны доопределениями, выполненными на итерации 1. Таким образом, для трех строк 1001, 1011, 1111, для которых отсутствовали условия доопределения (стояла пометка «Нет на итерации 1»), появляются новые условия доопределения и по ним составляется дополнительное логическое уравнение.

Таблица 11
Итерация 2. Шаг 2
Table 11
Iteration 2. Step 2

Строка таблицы истинности *Row of the truth table*		Разложение $f_3^* \prec (f_1^* \vee f_2^*)$ *Decomposition* $f_3^* \prec (f_1^* \vee f_2^*)$			Разложение $f_5^* \prec (f_2^* \& f_4^*)$ *Decomposition* $f_5^* \prec (f_2^* \& f_4^*)$			Разложение $f_7^* \prec (f_1^* \vee f_6^*)$ *Decomposition* $f_7^* \prec (f_1^* \vee f_6^*)$		
		f_1	f_2	f_3	f_2	f_4	f_5	f_1	f_6	f_7
1011	Значения функций	t_{21}	$t_{22}=1$	t_{23}						
	Условия доопределения	$t_{23}=1$			нет			нет		
	Уравнение	$t_{23}=1$								

Таблица 12
Итерация 2. Шаг 3
Table 12
Iteration 2. Step 3

Строка таблицы истинности *Row of the truth table*		Разложение $f_3^* \prec (f_1^* \vee f_2^*)$ *Decomposition* $f_3^* \prec (f_1^* \vee f_2^*)$			Разложение $f_5^* \prec (f_2^* \& f_4^*)$ *Decomposition* $f_5^* \prec (f_2^* \& f_4^*)$			Разложение $f_7^* \prec (f_1^* \vee f_6^*)$ *Decomposition* $f_7^* \prec (f_1^* \vee f_6^*)$		
		f_1	f_2	f_3	f_2	f_4	f_5	f_1	f_6	f_7
1111	Значения функций	$t_{36}=1$	t_{37}	t_{38}						
	Условия доопределения	$t_{38}=1$			нет			нет		
	Уравнение	$t_{38}=1$								

Уравнение, добавляемое после выполнения итерации 2, имеет следующий вид:

$$(\bar{t}_{13}\bar{t}_{14}\bar{t}_{15} \vee t_{13}\bar{t}_{14}t_{15})\, t_{23}\, t_{38} = 1.$$

Одно из решений этого уравнения – $t_{13}=0,\ t_{14}=0,\ t_{15}=0,\ t_{23}=1,\ t_{38}=1$.

Доопределение значения t_{13} логической переменной (значения функции f_1 на наборе 1001) вызывает необходимость доопределения переменных t_{17}, t_{18}, что приводит к итерации 3 (табл. 13).

Таблица 13
Итерация 3
Table 13
Iteration 3

Строка таблицы истинности *Row of the truth table*		Разложение $f_3^* \prec (f_1^* \vee f_2^*)$ *Decomposition* $f_3^* \prec (f_1^* \vee f_2^*)$			Разложение $f_5^* \prec (f_2^* \,\&\, f_4^*)$ *Decomposition* $f_5^* \prec (f_2^* \,\&\, f_4^*)$			Разложение $f_7^* \prec (f_1^* \vee f_6^*)$ *Decomposition* $f_7^* \prec (f_1^* \vee f_6^*)$		
		f_1	f_2	f_3	f_2	f_4	f_5	f_1	f_6	f_7
1001	Значения функций	$t_{13}=0$						$t_{13}=0$	t_{17}	t_{18}
		$t_{13}=1$						$t_{13}=1$		
	Условия доопределения	нет			нет			$((t_{13}=0)$ и $(t_{17}=0)$ и $(t_{18}=0))$ или $((t_{13}=0)$ и $(t_{17}=1)$ и $(t_{18}=1))$ или $((t_{13}=1)$ и $(t_{18}=1))$		
	Уравнение	$\bar{t}_{13}\bar{t}_{17}\bar{t}_{18} \vee \bar{t}_{13}t_{17}t_{18} \vee t_{13}t_{18}=1$								

Уравнение, добавляемое после выполнения итерации 3:

$$\bar{t}_{13}\bar{t}_{17}\bar{t}_{18} \vee \bar{t}_{13}t_{17}t_{18} \vee t_{13}t_{18}=1.$$

Одно из решений этого уравнения – $t_{13}=0,\ t_{17}=0,\ t_{18}=0$.

Результирующее логическое уравнение, составленное для проверки существования совместного разложения и нахождения доопределений функций, имеет вид

$$t_1(\bar{t}_1 \vee \bar{t}_2)t_3\ \bar{t}_4\ t_5\ \bar{t}_6\ t_7\ t_9(\bar{t}_8\bar{t}_{10} \vee t_8t_{10})t_{11}(t_{11} \vee t_{12})(\bar{t}_{14} \vee \bar{t}_{16})\bar{t}_{19}\ \bar{t}_{20}\ t_{22}\ t_{24}\ t_{25}\ t_{26}\ t_{27}\ t_{29}\ t_{35}\ t_{36}\ \&$$
$$\&\, t_{29}\ t_{35}\ t_{36}(\bar{t}_{13}\bar{t}_{14}\bar{t}_{15} \vee t_{13}\bar{t}_{14}t_{15})t_{23}\ t_{38}(\bar{t}_{13}\bar{t}_{17}\bar{t}_{18} \vee \bar{t}_{13}t_{17}t_{18} \vee t_{13}t_{18})=1. \qquad (11)$$

Методам решения логических уравнений посвящена монография [8]. Целесообразно привести левую часть уравнения к виду КНФ и получить задачу о выполнимости КНФ, которая имеет многочисленные применения при верификации описаний функционирования цифровых систем и которой посвящено большое число научных исследований [4]. Эффективные программы (SAT-solvers) решения логических уравнений используют КНФ в левых частях уравнений [5]. Приведем ДНФ-сомножители $\bar{t}_8\bar{t}_{10} \vee t_8t_{10}$, $\bar{t}_{13}\bar{t}_{14}\bar{t}_{15} \vee t_{13}\bar{t}_{14}t_{15}$, $\bar{t}_{13}\bar{t}_{17}\bar{t}_{18} \vee \bar{t}_{13}t_{17}t_{18} \vee t_{13}t_{18}$ к представлению в виде КНФ. Чтобы перейти от ДНФ D булевой функции к КНФ K, получим сначала инверсию $\bar{D}$ ДНФ D, упростим полученную ДНФ $\bar{D}$, затем воспользуемся *принципом двойственности* для получения КНФ $K=\bar{\bar{D}}$ [9, с. 20]. Например,

$$\bar{D}=\neg(\bar{t}_{13}\bar{t}_{17}\bar{t}_{18} \vee \bar{t}_{13}t_{17}t_{18} \vee t_{13}t_{18})=(t_{13} \vee t_{17} \vee t_{18})\&(t_{13} \vee \bar{t}_{17} \vee \bar{t}_{18})\&(\bar{t}_{13} \vee \bar{t}_{18})=$$
$$=t_{13} \vee t_{13}\bar{t}_{17} \vee t_{13}\bar{t}_{18} \vee t_{13}t_{17} \vee t_{17}\bar{t}_{18} \vee t_{13}t_{18} \vee \bar{t}_{17}t_{18};$$

$$K=\bar{\bar{D}}=(t_{13} \vee \bar{t}_{17} \vee t_{18})\&(t_{13} \vee t_{17} \vee \bar{t}_{18})\&(\bar{t}_{13} \vee t_{18})\&(\bar{t}_{13} \vee t_{17} \vee t_{18})\&(\bar{t}_{13} \vee \bar{t}_{17} \vee t_{18}).$$

Здесь символ $\neg$ обозначает инверсию выражения, перед которым он стоит. Полученная формула имеет вид КНФ. Аналогично получаем другие КНФ:

$$\bar{t}_8\bar{t}_{10} \vee t_8t_{10}=(t_8 \vee \bar{t}_{10})\&(\bar{t}_8 \vee t_{10}),$$

$$\bar{t}_{13}\bar{t}_{14}\bar{t}_{15} \vee t_{13}\bar{t}_{14}t_{15}=(\bar{t}_{13} \vee \bar{t}_{14})\&(\bar{t}_{13} \vee t_{15})\&(t_{13} \vee \bar{t}_{15})\&\bar{t}_{14}\&(\bar{t}_{17} \vee t_{15})\&(t_{13} \vee \bar{t}_{15})\&(\bar{t}_{17} \vee \bar{t}_{15}).$$

Запишем левую часть результирующего уравнения (11) в виде КНФ, в которой для сокращения записи оставлены только символы &, соединяющие строки:

$$t_1(\bar{t}_1 \vee \bar{t}_2)\, t_3\, \bar{t}_4\, t_5\, \bar{t}_6\, t_7\, t_9(\bar{t}_8\bar{t}_{10} \vee t_8 t_{10})\, t_{11}(t_{11} \vee t_{12})\, t_{12}(\bar{t}_{14} \vee \bar{t}_{16})\, \bar{t}_{19}\, \bar{t}_{20}\, t_{22}\, t_{24}\, t_{25}\, t_{26}\, t_{27}\, t_{29}\, t_{35}\, t_{36} \&$$
$$\&(\bar{t}_{13} \vee \bar{t}_{14})(\bar{t}_{13} \vee t_{15})(t_{13} \vee \bar{t}_{15})\, \bar{t}_{14}(\bar{t}_{17} \vee t_{15})(t_{13} \vee \bar{t}_{15})(\bar{t}_{17} \vee \bar{t}_{15})\, t_{23}\, t_{38} \&$$
$$\&(t_{13} \vee \bar{t}_{17} \vee t_{18}) \& (t_{13} \vee t_{17} \vee \bar{t}_{18}) \& (\bar{t}_{13} \vee t_{18}) \& (\bar{t}_{13} \vee t_{17} \vee t_{18}) \& (\bar{t}_{13} \vee \bar{t}_{17} \vee t_{18}) = 1. \qquad (12)$$

Утверждение 7. *Корень логического уравнения, выражающего условия существования совместного разложения, является доопределением значений функций, при котором существует совместное разложение.*

Утверждение 8. *Если логическое уравнение, выражающее условия существования совместного разложения, не имеет корней (решений), то совместное разложение не существует.*

Совместное разложение (9) существует, так как уравнение (12) имеет корни. Один из корней уравнения (12) выглядит следующим образом:

$$t_1=1,\ t_2=0,\ t_3=1,\ t_4=0,\ t_5=1,\ t_6=0,\ t_7=1,\ t_8=0,\ t_9=1,\ t_{10}=0,$$
$$t_{11}=1,\ t_{12}=1,\ t_{13}=0,\ t_{14}=0,\ t_{15}=0,\ t_{16}=0,\ t_{17}=0,\ t_{18}=0,\ t_{19}=0,\ t_{20}=0,$$
$$t_{22}=1,\ t_{23}=1,\ t_{24}=1,\ t_{25}=1,\ t_{26}=1,\ t_{27}=1,\ t_{29}=1,\ t_{35}=1,\ t_{36}=1,\ t_{38}=1.$$

Другие оставшиеся семь корней (всего корней восемь) уравнения (12) дадут другие доопределения базисных функций, при которых первое совместное разложение также будет существовать. Замены неопределенных значений функций найденными определенными значениями корней уравнения (12) показаны в табл. 14. Переменные t_{21}, t_{28}, t_{30}, t_{31}, t_{32}, t_{33}, t_{34}, t_{37}, t_{39}, t_{40} не участвовали в записи уравнения (12), поэтому соответствующие значения функций остались неопределенными (табл. 15). Сумма весов базисных функций f_1^*, f_2^*, f_4^*, f_6^* равна 13. Логическая схема (рис. 8) построена по первому совместному разложению.

Таблица 14
Замена доопределяемых значений функций из табл. 8 корнями уравнения (12)
Table 14
Replacing up to the predefined values of the functions from Table 8 with the roots of equation (12)

$x_1\ x_2\ x_3\ x_4$	f_1^*	f_2^*	f_3^*	f_4^*	f_5^*	f_6^*	f_7^*
0 0 0 0	–	–	–	–	–	–	–
0 0 0 1	0	$-(t_1=1)$	1	$-(t_2=0)$	0	1	$-(t_3=1)$
0 0 1 0	1	0	1	1	$-(t_4=0)$	0	$-(t_5=1)$
0 0 1 1	–	–	–	–	–	–	–
0 1 0 0	0	0	$-(t_6=0)$	1	0	$-(t_7=1)$	1
0 1 0 1	–	–	–	–	–	–	–
0 1 1 0	$-(t_8=0)$	1	$-(t_9=1)$	1	1	0	$-(t_{10}=0)$
0 1 1 1	–	–	–	–	–	–	–
1 0 0 0	$-(t_{11}=1)$	0	1	0	0	$-(t_{12}=1)$	1
1 0 0 1	$-(t_{13}=0)$	$-(t_{14}=0)$	$-(t_{15}=0)$	$-(t_{16}=0)$	0	$-(t_{17}=0)$	$-(t_{18}=0)$
1 0 1 0	0	0	0	0	$-(t_{19}=0)$	$-(t_{20}=0)$	0
1 0 1 1	$-(t_{21})$	$-(t_{22}=1)$	$-(t_{23}=1)$	$-(t_{24}=1)$	1	1	$-(t_{25}=1)$
1 1 0 0	–	–	–	–	–	–	–
1 1 0 1	1	1	$-(t_{26}=1)$	1	$-(t_{27}=1)$	$-(t_{28})$	$-(t_{29}=1)$
1 1 1 0	$-(t_{30})$	$-(t_{31})$	$-(t_{32})$	$-(t_{33})$	$-(t_{34})$	1	$-(t_{35}=1)$
1 1 1 1	$-(t_{36}=1)$	$-(t_{37})$	$-(t_{38}=1)$	$-(t_{39})$	$-(t_{40})$	0	1

Таблица 15
Доопределенные частичные функции для первого совместного разложения
Table 15
Predefined partial functions for the first joint decomposition

$x_1\ x_2\ x_3\ x_4$	f_1^*	f_2^*	f_3^*	f_4^*	f_5^*	f_6^*	f_7^*
0 0 0 0	–	–	–	–	–	–	–
0 0 0 1	0	1	1	0	0	1	1
0 0 1 0	1	0	1	1	0	0	1
0 0 1 1	–	–	–	–	–	–	–
0 1 0 0	0	0	0	1	0	1	1
0 1 0 1	–	–	–	–	–	–	–
0 1 1 0	0	1	1	1	1	0	0
0 1 1 1	–	–	–	–	–	–	–
1 0 0 0	1	0	1	0	0	1	1
1 0 0 1	0	0	0	0	0	0	0
1 0 1 0	0	0	0	0	0	0	0
1 0 1 1	–	1	1	1	1	1	1
1 1 0 0	–	–	–	–	–	–	–
1 1 0 1	1	1	1	1	1	–	1
1 1 1 0	–	–	–	–	–	1	1
1 1 1 1	1	–	1	–	–	0	1
Вес w_{f_i}	4	3	6	3	3	3	7

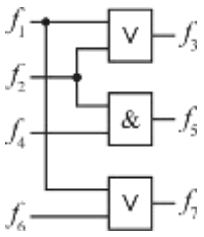


Рис. 8. Логическая схема для первого совместного разложения
Fig. 8. The logical circuit for the first joint decompositions

Второе совместное разложение. Проверим, существуют ли для трех заданных формул $f_3 = f_1 \vee f_2$, $f_5 = f_3 \,\&\, f_4$, $f_7 = f_5 \vee f_6$ такие доопределения (помечены символами **) частичных функций из табл. 3, чтобы выполнялось совместное разложение вида

$$f_3^{**} \prec (f_1^{**} \vee f_2^{**}),\ f_5^{**} \prec (f_3^{**} \,\&\, f_4^{**}),\ f_7^{**} \prec (f_5^{**} \vee f_6^{**}). \qquad (13)$$

Назовем это совместное разложение *вторым* совместным разложением. Такого вида разложения показаны на рис. 7, *а*. Граф уравнений, изображенный на рис. 7, *а*, изоморфен графу совместного разложения (11) (рис. 9), необходимо только в оба графа ввести дополнительные вершины, которые будут соответствовать функциям. Графы являются *изоморфными*, если они совпадают с точностью до обозначения вершин [10]. Заметим, что и графы G зависимости функций, построенные по этим графам уравнений, также будут изоморфными.

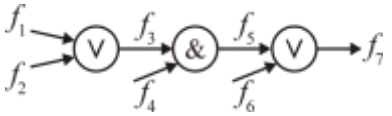


Рис. 9. Граф уравнений второго совместного разложения (11)
Fig. 9. Graph of equations of the second joint decomposition (11)

Результирующие доопределения функций, найденные путем составления и решения нового логического уравнения, заданы в табл. 16. Сумма доопределений базисных функций равна 15. Логическая схема второго совместного разложения изображена на рис. 10.

Таблица 16
Доопределенные частичные функции для второго совместного разложения
Table 16
Predefined partial functions for the second joint decomposition

$x_1\ x_2\ x_3\ x_4$	f_1^{**}	f_2^{**}	f_3^{**}	f_4^{**}	f_5^{**}	f_6^{**}	f_7^{**}
0 0 0 0	–	–	–	–	–	–	–
0 0 0 1	0	–(1)	1	–(0)	0	1	–(1)
0 0 1 0	1	0	1	1	–(1)	0	–(1)
0 0 1 1	–	–	–	–	–	–	–
0 1 0 0	0	0	–(0)	1	0	–(1)	1
0 1 0 1	–	–	–	–	–	–	–
0 1 1 0	–	1	–(1)	1	1	0	–(1)
0 1 1 1	–	–	–	–	–	–	–
1 0 0 0	–(1)	0	1	0	0	–(1)	1
1 0 0 1	–(0)	–(0)	–(0)	–(0)	0	–(0)	–(0)
1 0 1 0	0	0	0	0	–(0)	–(0)	0
1 0 1 1	–	–(1)	–(1)	–(1)	1	1	–(1)
1 1 0 0	–	–	–	–	–	–	–
1 1 0 1	1	1	–(1)	1	–(1)	–	–(1)
1 1 1 0	–	–	–	–	–	1	–(1)
1 1 1 1	–(1)	–(1)	–(1)	–(1)	–(1)	0	1
Вес w_{f_i}	3	4	6	4	4	4	7

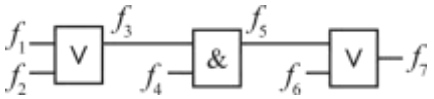


Рис. 10. Логическая схема для второго совместного разложения
Fig. 10. The logical circuit for the second joint decomposition

Эвристический алгоритм [3] поиска наибольшего множества представимых функций для системы полностью определенных кофакторов при поиске *множеств* совместных разложений может быть обощен на случай частичных кофакторов. Однако одной проверки на бесконтурность очередного подграфа графа G для частичных функций мало, так как для множества вершин очередного подграфа (множества кофакторов) должна выполняться процедура проверки существования общего доопределения, сводящаяся к решению логического уравнения. Достоинством такого алгоритма является то, что он не ограничивается числом вершин в бесконтурных подграфах и конкретными видами совместных разложений, примеры которых показаны на рис. 4–7. Обобщение алгоритма [3] для частичных функций требует отдельного изучения и в данной работе не рассматривается.

Синтез схем. После решения задачи 2 либо задачи 3 частичные базисные функции и те, которые не участвовали в разложениях, могут подвергаться дальнейшей логической оптимизации, например доопределению с целью поиска максимального по мощности подмножества взаимно инверсных частичных функций [11], либо подвергаться минимизации в классе ДНФ, после чего могут быть применены уже известные алгоритмы [3, 12] представления полностью определенных кофакторов в виде алгебраических разложений. Заметим, что процедура поиска взаимно инверсных частичных функций может быть выполнена и перед началом поиска дизъюнктивных и конъюнктивных разложений.

Проведем BDD-минимизацию как исходной системы кофакторов (см. табл. 3), так и систем базисных функций, полученных в результате двух раздельных и двух совместных разложений. Для этого воспользуемся:

– программой OPT_BDD [1, с. 489] BDD-минимизации систем полностью определенных функций, при этом исходные частичные базисные булевы функции доопределяются до полностью определенных путем «нулевого» доопределения (все неопределенные значения заменяются нулями);

– программой [1, с. 516], выполняющей доопределение частичных базисных функций до полностью определенных в процессе построения BDD; получающееся доопределение частичных функций в этом случае не является «нулевым».

Все результаты BDD-минимизации имеют вид многоуровневых представлений систем полностью определенных функций в виде взаимосвязанных логических уравнений, которые переводятся в синтезируемые VHDL-описания и по которым могут быть синтезированы логические схемы.

VHDL-описания для синтеза логических схем получались следующим образом: для каждого множества базисных функций проводилась BDD-минимизация как в классе полностью определенных, так и в классе частичных функций, затем добавлялись соответствующие формулы дизъюнктивных либо конъюнктивных разложений. Например, для второго множества раздельных разложений $E_9=\{ f_3^* \prec (f_4^* \vee f_7^*)\}$, $E_{14}=\{ f_5^* \prec (f_1^* \vee f_2^*)\}$ соответствующее многоуровневое представление, полученное программой [1, с. 516] оптимизации BDD-представлений, имеет вид

$$r_1 = \overline{x}_5 f_1 \vee x_5 f_2,\ r_2 = \overline{x}_5 f_2 \vee x_5 f_3,\ r_3 = \overline{x}_5 f_3 \vee x_5 f_5,\ r_4 = \overline{x}_5 f_4 \vee x_5 f_3,\ r_5 = \overline{x}_5 f_6 \vee x_5 f_7,$$
$$f_1 = \overline{x}_1 x_3 \vee x_1 s_1,\ f_2 = s_1,\ f_3 = f_4 \vee f_7,\ f_5 = f_1 \vee f_2,\ f_4 = \overline{x}_1 \vee x_1 s_1,\ f_6 = \overline{x}_1 s_2 \vee x_1 s_1,$$
$$f_7 = \overline{x}_1 \vee x_1 s_2,\ s_1 = \overline{x}_3 s_3 \vee x_3 s_4,\ s_2 = \overline{x}_3 \vee x_3 s_3,\ s_3 = x_2 x_4;\ s_4 = \overline{x}_2 x_4 \vee x_2 \overline{x}_4$$

и показано на рис. 11.

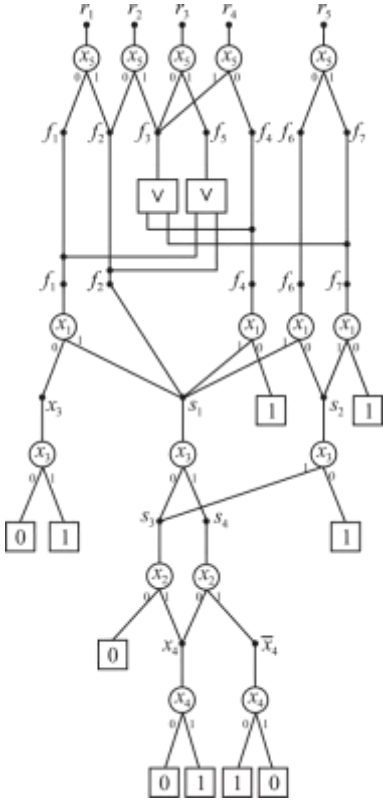


Рис. 11. Многоуровневое представление полностью определенных булевых функций r_1, r_2, r_3, r_4, r_5

Fig. 11. Multilevel representation of fully defined Boolean functions r_1, r_2, r_3, r_4, r_5

Логические уравнения конвертировались в VHDL-описания, подаваемые на вход синтезатора LeonardoSpectrum, в котором выполнялся синтез логических схем в библиотеке [1, с. 448] проектирования заказных КМОП СБИС. Результаты синтеза представлены в табл. 17, где *Area* – суммарная площадь элементов схемы (в условных единицах), *Delay* – временна́я задержка схемы (нс). Символом * помечены меньшие значения площади и задержки.

Таблица 17
Результаты синтеза схем
Table 17
Results of circuit synthesis

Функциональное описание *Functional description*	BDD-минимизация в классе полностью определенных булевых функций *BDD-minimization in a class of fully defined Boolean functions*		BDD-минимизация в классе частичных булевых функций *BDD-minimization in the class of partial Boolean functions*	
	Area	*Delay*	*Area*	*Delay*
Кофакторы (см. табл. 3)	9 213	2,31	5 781	2,32
Функции первого множества раздельных разложений	9 185	2,95	6 891	2,93
Функции второго множества раздельных разложений	9 347	2,28	*5 491	2,80
Функции первого совместного разложения	9 871	2,98	5 971	*2,16
Функции второго совместного разложения	8 710	3,73	8 158	3,33

Анализ результатов синтеза показывает, что BDD-минимизация частичных функций имеет преимущества по сравнению с BDD-минимизацией в классе полностью определенных функций, а нахождение множеств раздельных либо совместных разложений (дизъюнктивных, конъюнктивных) может улучшать схемные реализации по сравнению с BDD-минимизацией, которая проводится без учета дизъюнктивных и конъюнктивных разложений кофакторов функций. Эти результаты показывают также, что целесообразно развивать методы алгебраических разложений частичных функций при BDD-минимизации, а не переходить с помощью «нулевого доопределения» к полностью определенным функциям и для них выполнять BDD-оптимизацию, как это обычно принято в литературе.

Заключение. В статье предложены приближенные алгоритмы построения дизъюнктивных и конъюнктивных разложений систем частичных булевых функций – кофакторов одного уровня бинарной диаграммы решений. Алгоритмы могут быть обобщены для других видов алгебраических разложений, когда выходными функциями разложений являются не только дизъюнкции и конъюнкции, но и отрицания данных логических операций. Применение предложенных алгоритмов и уже известных алгоритмов минимизации многоуровневых BDD-представлений систем частичных функций позволяет получать лучшие результаты технологически независимой логической оптимизации – начального этапа синтеза многовыходных логических схем в базисе библиотечных элементов либо в базисе программируемых логических интегральных схем.

Список использованных источников

1. Бибило, П. Н. Бинарные диаграммы решений в логическом проектировании / П. Н. Бибило. – М. : Ленанд, 2024. – 560 с.

2. Yang, S. BDS: a BDD-based logic optimization system / S. Yang, M. Ciesielski // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2002. – Vol. 21, no. 7. – P. 866–876.

3. Бибило, П. Н. Экспериментальное исследование алгоритмов минимизации BDDI-представлений систем булевых функций с использованием алгебраических разложений кофакторов / П. Н. Бибило, В. И. Романов // Программная инженерия. – 2022. – Т. 13, № 2. – С. 51–67.

4. Handbook of Satisfiability / ed.: A. Biere, M. Heule, H. van Maaren, T. Valsh. – IOS Press, 2009. – 980 p.

5. Goldberg, E. BerkMin: a fast and robust sat-solver / E. Goldberg, Y. Novikov // Discrete Applied Mathematics. – 2007. – Vol. 155, no. 12. – P. 1549–1561.

6. Закревский, А. Д. Логический синтез каскадных схем / А. Д. Закревский. – М. : Наука, 1981. – 416 с.

7. Еремеев, А. В. Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования / А. В. Еремеев, Л. А. Заозерская, А. А. Колоколов // Дискретный анализ и исследование операций. – 2000. – Т. 7, вып. 2. – С. 22–46.

8. Закревский, А. Д. Логические уравнения / А. Д. Закревский. – Минск : Наука и техника, 1975. – 96 с.

9. Бибило, П. Н. Декомпозиция булевых функций на основе решения логических уравнений / П. Н. Бибило. – Минск : Беларус. навука, 2009. – 211 с.

10. Лекции по теории графов / В. А. Емеличев, О. И. Мельников, В. И. Сарванов, Р. И. Тышкевич. – М. : Наука, 1990. – 384 с.

11. Бибило, П. Н. Минимизация BDDI-представлений систем не полностью определенных булевых функций / П. Н. Бибило // Программная инженерия. – 2020. – Т. 11, № 3. – С. 152–168.

12. Бибило, П. Н. Минимизация многоуровневых представлений систем полностью определенных булевых функций с использованием разложений Шеннона и алгебраических представлений кофакторов / П. Н. Бибило, В. И. Романов // Информатика. – 2021. – Т. 18, № 2. – С. 7−32.

13. Брейтон, Р. К. Синтез многоуровневых комбинационных логических схем / Р. К. Брейтон, Г. Д. Хэчтел, А. Л. Санджованни-Винчентелли // Труды Института инженеров по электротехнике и радиоэлектронике. – 1990. – Т. 78, № 2. – С. 38–83.

References

1. Bibilo P. N. Binarnye diagrammy reshenij v logicheskom proektirovanii. *Binary Decision Diagrams in Logical Design*. Moscow, Lenand, 2024, 560 p. (In Russ.).

2. Yang S., Ciesielski M. BDS: a BDD-based logic optimization system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2002, vol. 21, no. 7, pp. 866–876.

3. Bibilo P. N., Romanov V. I. *Experimental study of algorithms for minimization of binary decision diagrams using algebraic representations of cofactors*. Programmnaya ingeneria [*Software Engineering*], 2022, vol. 13, no. 2, pp. 51–67 (In Russ.).

4. *Handbook of Satisfiability*. In A. Biere, M. Heule, H. van Maaren, T. Valsh (eds.). IOS Press, 2009, 980 p.

5. Goldberg E., Novikov Y. BerkMin: a fast and robust SAT-solver. *Discrete Applied Mathematics*, 2007, vol. 155, no. 12, pp. 1549–1561.

6. Zakrevskij A. D. Logicheskij sintez kaskadnyh skhem. *Logical Synthesis of Cascading Circuit.* Moscow, Nauka, 1981, 416 p. (In Russ.).

7. Eremeev A. V., Zaozerskaya L. A., Kolokolov A. A. *The problem of covering the set: complexity, algorithms, experimental studies*. Diskretnyj analiz i issledovanie operacij [*Discrete Analysis and Operations Research*], 2000, vol. 7, no. 2, pp. 22–46 (In Russ.).

8. Zakrevskij A. D. Logicheskie uravneniya. *Logical Equations*. Minsk, Nauka i tekhnika, 1975, 96 p. (In Russ.).

9. Bibilo P. N. Dekompoziciya bulevyh funkcij na osnove resheniya logicheskih uravnenij. *Decomposition of Boolean Functions Based on Solving Logical Equations*. Minsk, Belaruskaja navuka, 2009, 211 p. (in Russ.).

10. Emelichev V. A., Mel'nikov O. I., Sarvanov V. I., Tyshkevich R. I. Lekcii po teorii grafov. *Lectures on Graph Theory*. Moscow, Nauka, 1990, 384 p. (In Russ.).

11. Bibilo P. N. *Minimization of binary decision diagrams for systems of incompletely defined Boolean functions using inverse cofactors*. Programmnaya ingeneria [*Software Engineering*], 2020, vol. 11, no. 3, pp. 152–168 (In Russ.).

12. Bibilo P. N., Romanov V. I. *Minimization of binary decision diagrams for systems of completely defined Boolean functions using Shannon expansions and algebraic representations of cofactors*. Informatika [*Informatics*], 2021, vol. 18, no. 2, pp. 7–32 (In Russ.).

13. Brayton R. K., Hachtel G. D., Sangiovanni-Vincentelli A. L. *Synthesis of multilevel combinational logic circuits*. Trudy Instituta inzhenerov po jelektrotehnike i radiojelektronike [*Proceedings of the IEEE*], 1990, vol. 78, no. 2, pp. 38–83 (In Russ.).

Информация об авторе

Бибило Петр Николаевич, доктор технических наук, профессор, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: bibilo@newman.bas-net.by

Information about the author

Petr N. Bibilo, D. Sc. (Eng.), Prof., The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: bibilo@newman.bas-net.by

ОБРАБОТКА СИГНАЛОВ, ИЗОБРАЖЕНИЙ, РЕЧИ, ТЕКСТА И РАСПОЗНАВАНИЕ ОБРАЗОВ

SIGNAL, IMAGE, SPEECH, TEXT PROCESSING AND PATTERN RECOGNITION

УДК 004
DOI: 10.37661/1816-0301-2025-22-1-66-72

Оригинальная статья
Original Article

Сравнительный анализ алгоритмов отслеживания объектов

Б. А. Залесский✉, В. А. Иванюкович, К. В. Реер, Д. А. Старикович

Объединенный институт проблем информатики
Национальной академии наук Беларуси,
ул. Сурганова, 6, Минск, 220012, Беларусь
✉*E-mail: zalesky@newman.bas-net.by*

Аннотация

Цели. Представляются результаты вычисления и сравнительного анализа характеристик нейросетевого алгоритма отслеживания объекта (трекера), предложенного авторами в работе [1], с двумя другими алгоритмами при решении актуальной задачи автоматического обнаружения и дальнейшего сопровождения дронов. Для сравнительного анализа были выбраны: ByteTrack – один из лучших в настоящее время среди трекеров открытого доступа и простой трекер, основанный на использовании нейросетевого детектора и корреляции вместе с фильтром Калмана. Первый из трекеров был выбран потому, что он может быть реализован на языке C++ без применения сторонних библиотек и фреймворков и использован на малых вычислителях в режиме реального времени. Второй трекер тестировался для выяснения того, насколько новые трекеры лучше простых хорошо известных. Особенность используемых алгоритмов заключается в автоматическом обнаружении и захвате дрона, его дальнейшем надежном сопровождении, быстром повторном захвате в случае срыва сопровождения, захвате другого дрона при исчезновении сопровождаемого объекта. В анализируемых трекерах обнаружение дрона на кадрах видео осуществляется с помощью нейронной сети-детектора, а сопровождение – с помощью нейронной сети-детектора и разработанных алгоритмов.

Методы. Для проведения сравнительного анализа алгоритмов отслеживания объекта были созданы и размечены два датасета. Они представляют собой видео, на кадрах которых присутствуют дроны разных типов. Первый из датасетов, содержащий 36 895 кадров, использовался для обучения алгоритмов, а второй, состоящий из 8678 кадров, – для вычисления характеристик алгоритмов и выполнения сравнительного анализа. Видео обучающего и тестового датасетов сняты разными камерами в различных условиях. Для обучения нейросетевой части трекеров были написаны версии алгоритмов на языке программирования Python, а для вычисления и анализа характеристик в условиях, близких к реальным, – на языке C++, что потребовало конвертации обученной сети с помощью фреймворка TensorRT. Также были реализованы программные средства сбора и обработки экспериментальных данных.

Результаты. Проведенный сравнительный анализ трех алгоритмов отслеживания объекта позволил вычислить и сравнить их характеристики, а также сделать выводы о способе обучения использованной

нейронной сети-детектора; о возможности применения трекеров в режиме реального времени на бюджетных персональных компьютерах с бюджетными видеокартами, имеющими программно-аппаратную архитектуру CUDA, и о применимости двух из них для решения задачи практического отслеживания дронов, наблюдаемых видеокамерами, с достаточной точностью и надежностью. Из трех протестированных алгоритмов наилучшие характеристики имеет разработанный авторами.
Заключение. Проведенный сравнительный анализ трекеров показал возможность практического применения трекера и алгоритма ByteTrack для решения задачи отслеживания дронов, однако в настоящее время сохраняется проблема обнаружения малоразмерных беспилотных летательных аппаратов.

Ключевые слова: видео, детекторы объектов, нейронные сети, алгоритмы отслеживания, дроны, характеристики трекеров, сравнительный анализ

Для цитирования. Сравнительный анализ алгоритмов отслеживания объектов / Б. А. Залесский, В. А. Иванюкович, К. В. Реер, Д. А. Старикович // Информатика. – 2025. − Т. 22, № 1. – С. 66–72. – DOI: 10.37661/1816-0301-2025-22-1-66-72.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 05.02.2025
Подписана в печать | Accepted 21.02.2025
Опубликована | Published 31.03.2025

Comparative analysis of object tracking algorithms

Boris A. Zalesky✉, Vladimir A. Ivanyukovich, Kiril V. Reer, Danila A. Starikovich

The United Institute of Informatics Problems
of the National Academy of Sciences of Belarus,
st. Surganova, 6, Minsk, 220012, Belarus
✉*E-mail: zalesky@newman.bas-net.by*

Abstract
Objectives. The article presents the results of calculation and comparative analysis of the characteristics of the algorithm proposed by the authors in [1] for tracking an object captured by a video camera, when solving the urgent task of automatic detection and tracking of drones. Two algorithms were selected for comparative analysis, one of which is the currently known open source ByteTrack tracker, and the other is a simple tracker based on the use of the neural network, correlation comparison together with Kalman filter. The first tracker was chosen because it can be implemented in C++ without using third-party libraries and frameworks and used on small computers in real time. The second tracker was used to determine how much better new trackers are than simple, long-used ones. The specificity of the used algorithms is automatic detection and capture of the drone, its further reliable tracking, quick repeated capture in case of tracking failure, capture of another drone when the tracked object disappears. In the used trackers, drone detection in video frames is carried out using a neural network detector, and tracking is done with the help of the neural network detector and developed tracking algorithms.
Methods. To perform a comparative analysis of object tracking algorithms, two datasets consisting of video frames that contain drone images were created and labeled. The training dataset consists of 36895 frames whereas testing one contains 8678 images. The videos of the training and test datasets were shot with different cameras in different conditions. To train the neural network part of the trackers, versions of the algorithms were written in the Python programming language, and to calculate and analyze characteristics in conditions close to real ones, in C++, which required converting the trained network using the TensorRT framework. Software tools for gathering and processing experimental data were also implemented.
Results. The comparative analysis of three object tracking algorithms allowed us to calculate and compare the characteristics of these trackers, as well as draw conclusions about the method of training the used neural network detector; about the possibility of using trackers in real time on budget personal computers with budget

video cards that have the CUDA software and hardware architecture, about the applicability of two of them for solving the problem of practical tracking of drones observed by video cameras with sufficient accuracy and reliability. Of the three tested algorithms the tracker previously developed by the authors has the best characteristics.
Conclusion. The comparative analysis of the above-mentioned trackers showed the possibility of practical application of the tracker and the ByteTrack algorithm for solving the problem of tracking drones, however, there is still a problem with detecting small-sized unmanned aerial vehicles.

Keywords: video, object detectors, neural networks, tracking algorithms, drones, tracker characteristics, comparative analysis

For citation. Zalesky B. A., Ivanyukovich V. A., Reer K. V., Starikovich D. A. *Comparative analysis of object tracking algorithms.* Informatika [*Informatics*], 2025, vol. 22, no. 1, pp. 66−72 (In Russ.). DOI: 10.37661/1816-0301-2025-22-1-66-72.

Conflict of interest. The authors declare of no conflict of interest.

Введение. Задача автоматического обнаружения объекта заданного типа, наблюдаемого видеокамерой, и дальнейшего его сопровождения находит применение в широком спектре приложений, таких как мониторинг безопасности движения транспортных средств, медицинская диагностика, навигация роботов и автономного транспорта, оценка позы и походки человека, анализ активности диких животных, пешеходов и т. д. В подавляющем большинстве случаев наблюдаются движущиеся объекты движущейся видеокамерой. В настоящее время по данной тематике опубликовано значительное число работ, ссылки на которые можно найти, например, в работах[1] [2, 3], доступно множество программных средств[2, 3], в том числе экспериментальных. Часть опубликованных программных средств успешно применяется на практике. Особенностью большинства опубликованных алгоритмов и разработанных программных средств является их использование для трекинга нейронных сетей.

В настоящей работе приводятся результаты сравнительного анализа характеристик нейросетевого алгоритма отслеживания трекера [1], который для краткости будем называть OFT (Optical Flow Tracker), и двух других алгоритмов при решении актуальной задачи автоматического обнаружения и дальнейшего сопровождения дронов. Задача отслеживания объекта в данном случае понимается следующим образом: алгоритм автоматически обнаруживает и локализует положения дронов на кадрах видео, выбирает по заранее заданному критерию (например, по уровню доверия (confidence) обнаруженных объектов, а также по размеру изображения объекта и (или) его расположению на кадре видео) один из них и выполняет его сопровождение. При срыве сопровождения, вызванного исчезновением объекта с кадра или другими причинами, алгоритм перезапускается для захвата и сопровождения нового объекта заданного типа.

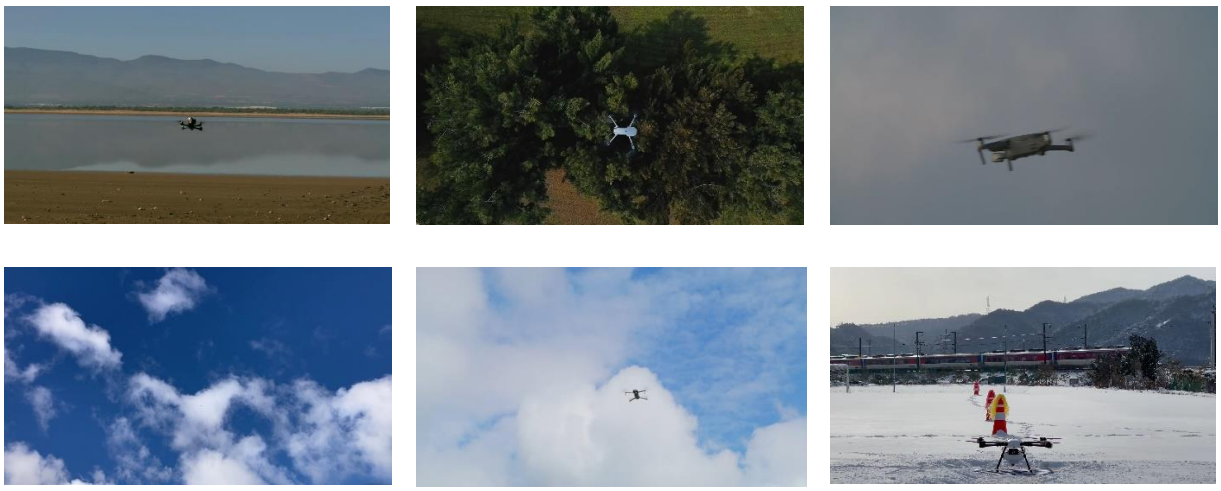
Одной из целей работы является сравнение характеристик нейросетевого трекера OFT с одними из лучших в настоящее время трекеров DeepSort и ByteTrack, имеющих открытые программные реализации, которые используются для решения практических задач отслеживания объектов. Эксперименты проводились на тестовых наборах видео (датасетах) дронов разных типов. Для тестирования были выбраны видео, характеристики которых отличаются от характеристик обучающих видео. Помимо разработанного алгоритма и трекеров DeepSort и ByteTrack тестировался относительно простой алгоритм, основанный на использовании нейросетевого детектора Yolov10 и корреляционного поиска (вместе с фильтром Калмана) на текущем кадре объекта, найденного на предыдущем кадре. Проведенное сравнение позволило сделать оценку, на сколько сложные современные алгоритмы лучше прежних простых при отслеживании дронов.

Заметим также, что алгоритмы отслеживания могут применяться в качестве детекторов. Использование результатов отслеживания для детекции в большинстве случаев ощутимо повыша-

[1]Object Tracking. – URL: https://paperswithcode.com/task/object-tracking (date of access: 24.01.2025).
[2]PyTorch. – URL: https://pytorch.org/vision/stable/index.html (date of access: 05.01.2024).
[3]Ultralytics YOLO Docs. – URL: https://docs.ultralytics.com/ru/models/yolo11/ (date of access: 05.01.2024).

ет ее точность и надежность. В статье приводятся результаты сравнения обнаружения объекта детектором и разработанным трекером.

Сравнительный анализ точности и надежности трекеров. Сравнительный анализ точности и надежности трекеров проводился на датасете, сформированном, аугментированном и размеченном авторами, так как не удалось найти в открытом доступе датасеты, содержащие дроны. При этом аугментировался только обучающий датасет, в котором количество кадров было доведено до 36 895. Тестовый набор данных состоял из 8678 кадров. Особенность обоих датасетов – наличие в них большого числа изображений дронов малых размеров. На рисунке показаны кадры из созданного датасета.



Примеры кадров из тестового датасета, содержащих дрон
Examples of frames from the test dataset containing a drone

На начальном этапе сравнительного анализа было установлено, что, несмотря на опубликованные результаты сравнения трекеров DeepSort и ByteTrack, в которых доказывалось превосходство DeepSort над ByteTrack, в экспериментах на созданных авторами датасетах ByteTrack показал лучшие результаты, поэтому в дальнейшем только он был использован для сравнительного анализа.

Ранее было проведено тестирование трекера OFT, реализованного на языке Python. Результаты тестирования приведены в работе [1], где для сравнительного анализа использовалась версия трекера ByteTrack, реализованная на Python, и двух трекеров – OFT и простого корреляционного, реализованных на C++. Для этого нейросетевой детектор Yolov10, реализованный на Python, был обучен и затем конвертирован с помощью фреймворка TensorRT для применения в C++-приложении. Важно отметить, что такая конвертация снизила значения характеристик Accuracy и Recall на тестовых видео на 1–3 % (эффект снижения характеристик нейросетевых детекторов объектов при конвертации наблюдался и на других видео).

При проведении сравнительного анализа оказалось, что нейросетевой детектор Yolov10 недостаточно хорошо обучается на созданном авторами и аугментированном датасете, поэтому в статье приводятся результаты, полученные после обучения детектора только на созданном авторами датасете, и результаты, полученные после дообучения детектора, предварительно обученного на датасете COCO, не содержащем изображения дронов.

Тестовый датасет был образован кадрами девяти видео, снятых в разных условиях. На видео были запечатлены квадрокоптеры на разном удалении от камеры. В табл. 1–3 приведены средние результаты, полученные на всем тестовом датасете для обученного в разной степени детектора Yolov10.

Во втором столбце таблиц даны результаты обнаружения объектов обученным авторами детектором Yolov10 (в процентах). Числовые данные в остальных столбцах показывают, на

сколько процентов результат применения каждого трекера улучшает соответствующий показатель детектора. В табл. 1 приведены результаты использования детектора Yolov10, обученного только на созданном авторами датасете.

Таблица 1
Результаты тестирования трекеров, обученных только на датасете, созданном авторами

Table 1
Test results of trackers trained only on the dataset created by the authors

Характеристики *Characteristics*	Детектор Yolov10, % *Detector Yolov10, %*	Трекер OFT, % *OFT Tracker, %*	Трекер корреляционный, % *Correlation tracker, %*
Accuracy	36,55	+12,29	+12,18
Recall	38,72	+12,51	+12,36

В табл. 2 приведены результаты, полученные при использовании детектора Yolov10, предварительно обученного на датасете COCO и дообученного на созданном датасете.

Таблица 2
Результаты тестирования трекеров, обученных на COCO и после этого дообученных

Table 2
Test results of trackers trained on COCO and after retraining

Характеристики *Characteristics*	Детектор Yolov10, % *Detector Yolov10, %*	Трекер OFT, % *OFT Tracker, %*	Трекер корреляционный, % *Correlation tracker, %*
Accuracy	52,37	+6,73	+4,14
Recall	53,99	+7,49	+6,48

Результаты, полученные для ByteTrack, приводятся отдельно, так как была использована его программная реализация на языке Python в отличие от программных реализаций трекера OFT и корреляционного трекера, написанных на языке C++ (напомним, что при конвертации детектора Yolov10 с языка Python на язык C++ его характеристики ухудшаются на 1–3 %).

Таблица 3
Результаты тестирования ByteTrack, реализованного на Python

Table 3
Test results of the ByteTrack implemented in Python

Характеристики *Characteristics*	ByteTrack, созданный датасет *ByteTrack, created dataset*	ByteTrack, COCO, созданный датасет *ByteTrack, COCO, created dataset*
Accuracy	+7,03	+7,63
Recall	+7,53	+8,37

Результаты, приведенные во втором столбце табл. 3, необходимо сравнивать с результатами третьего и четвертого столбцов табл. 1, а результаты третьего столбца табл. 3 – с результатами третьего и четвертого столбцов табл. 2. Полученные данные позволяют сделать вывод о том, что при использовании трекеров с нейросетевым детектором Yolov10, обученным только на датасете, созданном авторами, трекер OFT превосходит ByteTrack по точности (Accuracy) на 5,26 %, а по чувствительности (Recall) – на 4,98 %.

При использовании трекеров с нейросетевым детектором Yolov10, обученным сначала на COCO, а затем дообученным на датасете, созданном авторами, OFT и ByteTrack показали примерно одинаковые результаты (в данном случае следует принять во внимание потерю точности ByteTrack при конвертации его программной реализации с языка Python на язык C++).

Следует заметить, что все использованные с статье трекеры работают в режиме реального времени, однако самым быстрым из них оказался трекер OFT.

Заключение. Проведенный сравнительный анализ трекеров показал возможность практического применения трекеров OFT и ByteTrack для решения задачи отслеживания дронов в режиме реального времени. Оба трекера обладают точностью и чувствительностью, достаточными для отслеживания летящих дронов. При использовании трекеров с нейросетевым детектором Yolov10, обученным на датасете, который содержит только изображения дронов, предложенный авторами алгоритм превзошел ByteTrack по точности на 5 % и по чувствительности приблизительно на 4 %. Однако при применении трекеров вместе с детектором Yolov10, обученным сначала на большом датасете COCO, а затем дообученным на меньшем, который содержит дроны, характеристики точности и чувствительности трекеров оказались одинаковыми.

Между тем следует заметить, что точность и чувствительность всех трех детекторов ощутимо падают при отслеживании дронов, изображения которых имеют слишком малый размер (меньший чем 40×40 пикселов).

Вклад авторов. *Б. А. Залесский* сформулировал задачу, разработал трекер совместно с В. В. Иванюковичем, подготовил текст статьи. *В. А. Иванюкович* реализовал и обучил OFT, создал один из двух использованных датасетов и протестировал на нем OFT и ByteTrack. *К. В. Реер* реализовал корреляционный трекер и выполнил тестирование всех использованных трекеров. *Д. А. Старикович* создал второй использованный датасет, реализовал OFT и корреляционный трекер на C++.

Список использованных источников

1. Залесский, Б. А. Алгоритм сопровождения объекта, наблюдаемого видеокамерой / Б. А. Залесский, В. А. Иванюкович // Доклады Национальной академии наук Беларуси. – 2024. – Т. 68, № 2. – С. 105–111.

2. Proc. of Conf. on Computer Vision and Pattern Recognition 2024 (CVPR 2024). – Seattle, 17–21 June 2024. – URL: https://cvpr.thecvf.com/Conferences/2024 (date of access: 24.01.2025).

3. Proc. of Conf. on Computer Vision and Pattern Recognition 2023 (CVPR 2023). – Vancouver, 18–22 June 2023. – URL: https://openaccess.thecvf.com/CVPR2023 (date of access: 24.01.2025).

References

1. Zalesky B. A., Ivanyukovich V. A. *Algorithm for tracking an object observed by a video camera*. Doklady Nacional′noj akademii nauk Belarusi [*Doklady of the National Academy of Sciences of Belarus*], 2024, vol. 68, no. 2, pp. 105–111 (In Russ.).

2. *Proceedings of Conference on Computer Vision and Pattern Recognition 2024 (CVPR 2024)*. Seattle, 17–21 June 2024. Available at: https://cvpr.thecvf.com/Conferences/2024 (accessed 24.01.2025).

3. *Proceedings of Conference on Computer Vision and Pattern Recognition 2023 (CVPR 2023)*. Vancouver, 18–22 June 2023. Available at: https://openaccess.thecvf.com/CVPR2023 (accessed 24.01.2025).

Информация об авторах

Залесский Борис Андреевич, доктор физико-математических наук, Объединенный институт проблем информатики Национальной академии наук Беларуси.
E-mail: zalesky@newman.bas-net.by

Information about the authors

Boris A. Zalesky, D. Sc. (Eng.), The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.
E-mail: zalesky@newman.bas-net.by

Иванюкович Владимир Александрович, стажер младшего научного сотрудника, Объединенный институт проблем информатики Национальной академии наук Беларуси.

Реер Кирилл Васильевич, стажер младшего научного сотрудника, Объединенный институт проблем информатики Национальной академии наук Беларуси.

Старикович Данила Александрович, стажер младшего научного сотрудника, Объединенный институт проблем информатики Национальной академии наук Беларуси.

Vladimir A. Ivanyukovich, Trainee of Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.

Kirill V. Reer, Trainee of Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.

Danila A. Starikovich, Trainee of Junior Researcher, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus.

ЗАЩИТА ИНФОРМАЦИИ И НАДЕЖНОСТЬ СИСТЕМ
INFORMATION PROTECTION AND SYSTEM RELIABILITY

УДК 004.832.32
DOI: 10.37661/1816-0301-2025-22-1-73-89

Оригинальная статья
Original Article

Исследование физически неклонируемой функции конфигурируемого кольцевого осциллятора

А. А. Иванюк

Белорусский государственный университет информатики и радиоэлектроники,
ул. П. Бровки, 6, Минск, 220013, Беларусь
E-mail: ivaniuk@bsuir.by

Аннотация

Цели. Целью работы являются рассмотрение особенностей проектирования и реализации физически неклонируемой функции (ФНФ) конфигурируемого кольцевого осциллятора (ККО) на программируемых логических интегральных схемах (ПЛИС) типа FPGA и оценка основных параметров схем ККО и характеристик ФНФ ККО в различных сценариях моделирования и размещения на кристаллах FPGA.

Методы. Используются методы синтеза и анализа цифровых устройств, в том числе на программируемых логических интегральных схемах, основы цифровой схемотехники.

Результаты. Предложена обобщенная модель ФНФ, основанная на сравнении задержек распространения сигналов по паре симметричных путей. Модель включает в себя четыре основные стадии: генерирование множества симметричных путей (*Generate*), выборку из множества пары путей (*Select*/*Switch*), измерение задержки распространения сигнала для каждого выбранного пути (*Measure*) и вычисление бинарного ответа ФНФ на основе знака разницы измеренных задержек (*Compute*). Данная модель применима к таким классическим типам ФНФ, как ФНФ типа арбитр и ФНФ кольцевого осциллятора, и к их модификациям. На основе предложенной модели спроектирована ФНФ ККО, которая была реализована на ПЛИС типа FPGA Xilinx ZYNQ 7000. В ходе проведенных экспериментов над моделями и реализованными схемами были оценены основные временные параметры ККО и характеристики ФНФ ККО в различных сценариях моделирования и для двух типов размещения их компонент на кристаллах FPGA. Было показано, что подавляющую часть задержки распространения сигнала по выбранному пути составляет задержка на конфигурируемых межсоединениях FPGA, которая вне зависимости от типа применяемого размещения приводит к реализации множества заведомо асимметричных путей. Нарушение симметрии путей негативно сказывается на одной из важнейших характеристик ФНФ – внутрикристальной уникальности, низкие показатели которой могут служить сильным ограничением при реализации схем неклонируемой идентификации. При этом другие характеристики ФНФ, такие как единообразие, стабильность, надежность и внутрикристальная уникальность, имеют приемлемо высокие показатели.

Заключение. Проведенное параметрическое моделирование схем ФНФ ККО показало свою состоятельность при оценке таких характеристик ФНФ, как единообразие и внутрикристальная уникальность, что может быть использовано разработчиками для быстрой оценки качества проектируемых схем, не прибегая к их реализации. Обеспечение приемлемых значений межкристальной уникальности требует поиска новых схемотехнических решений, которые будут приводить к генерированию множества симметричных путей на ПЛИС типа FPGA. Кроме того, измеренные периоды схем ККО наглядно демон-

стрируют свою уникальность при их реализации как на одном, так и на различных кристаллах, что является основой для поиска новых методов и алгоритмов вычисления уникальных ответов ФНФ.

Ключевые слова: физическая криптография, физически неклонируемые функции, конфигурируемый кольцевой осциллятор, программируемые логические интегральные схемы, симметрия путей

Благодарности. Автор выражает искреннюю благодарность резиденту ПВТ компании «Инженерный Центр Ядро», которая является одним из центров разработки YADRO, за предоставленное оборудование для проведения экспериментов в рамках работы совместной учебной лаборатории с Белорусским государственным университетом информатики и радиоэлектроники.

Для цитирования. Иванюк, А. А. Исследование физически неклонируемой функции конфигурируемого кольцевого осциллятора / А. А. Иванюк // Информатика. – 2025. − Т. 22, № 1. – С. 73–89. – DOI: 10.37661/1816-0301-2025-22-1-73-89.

Конфликт интересов. Автор заявляет об отсутствии конфликта интересов.

Поступила в редакцию | Received 24.01.2025
Подписана в печать | Accepted 07.02.2025
Опубликована | Published 31.03.2025

Investigation of the physically unclonable function of a configurable ring oscillator

Alexander A. Ivaniuk

Belarusian State University
of Informatics and Radioelectronics,
st. P. Brovki, 6, Minsk, 220013, Belarus
E-mail: ivaniuk@bsuir.by

Abstract

Objectives. Design and implementation features of a Physically Unclonable Function (PUF) based on a Configurable Ring Oscillator (CRO) on FPGA platforms are examined. The study aims to evaluate the key parameters of CRO circuits and the characteristics of CRO-based PUFs under various simulation scenarios and placement configurations on FPGA dies.

Methods. Methods of synthesis and analysis of digital devices are employed, including those based on programmable logic integrated circuits, as well as the fundamentals of digital circuit design.

Results. A generalized model of a PUF based on the comparison of signal propagation delays along a pair of symmetric paths is proposed. The model includes four main stages: *Generate* – generating a set of symmetric paths, *Select/Switch* – selecting a pair of paths from the set, *Measure* – measuring the signal propagation delay for each selected path, and *Compute* – calculating the binary PUF response based on the sign of the difference between the measured delays. This model is applicable to classical PUF types such as Arbiter PUFs and Ring Oscillator PUFs, as well as their modifications. Based on the proposed model, a CRO PUF was designed and implemented on Xilinx ZYNQ 7000 FPGA devices. Experiments were conducted on both simulated models and implemented circuits to evaluate the key timing parameters of CROs and the characteristics of CRO PUFs under various simulation scenarios and two types of component placement on FPGA dies. The results demonstrated that the majority of the signal propagation delay along the selected path is determined by the delay in FPGA configurable interconnects. Regardless of the type of component placement used, this leads to the realization of predominantly asymmetric paths. The asymmetry in paths negatively impacts one of the most critical characteristics of PUFs – intra-chip uniqueness. Low intra-chip uniqueness can impose significant limitations when implementing unclonable identification schemes. However, other PUF characteristics, such as uniformity, stability, reliability, and inter-chip uniqueness, exhibited acceptably high-performance levels.

Conclusion. The conducted parametric simulation of CRO PUF circuits demonstrated its effectiveness in evaluating key PUF characteristics, such as uniformity and intra-chip uniqueness. This approach can be utilized by developers for rapid assessment of circuit quality without requiring physical implementation. Achieving acceptable inter-chip uniqueness values necessitates the development of new circuit design solutions that enable the generation of multiple symmetric paths on FPGA devices. Additionally, the measured periods of CRO circuits clearly demonstrate their uniqueness, both when implemented on the same chip and across different chips. This serves as a foundation for exploring new methods and algorithms for calculating unique PUF responses.

Keywords: physical cryptography, physically unclonable functions, configurable ring oscillator, programmable logic integrated circuits, symmetry of paths

Acknowledgments. The author expresses sincere gratitude to the HTP resident company "Engineering Center Yadro", which is one of the YADRO development centers, for providing equipment for conducting experiments as part of the work of a joint educational laboratory with the Belarusian State University of Informatics and Radioelectronics.

For citation. Ivaniuk A. A. *Investigation of the physically unclonable function of a configurable ring oscillator*. Informatika [*Informatics*], 2025, vol. 22, no. 1, pp. 73−89 (In Russ.). DOI: 10.37661/1816-0301-2025-22-1-73-89.

Conflict of interest. The author declares of no conflict of interest.

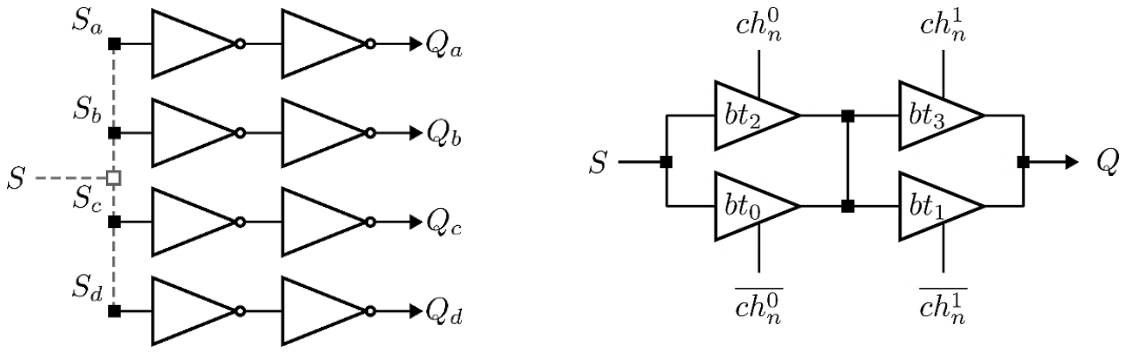
Введение. ФНФ, основанные на извлечении и измерении уникальных физических характеристик полупроводниковых кристаллов цифровых устройств, являются базовыми элементами физической криптографии [1]. К основным областям применения ФНФ можно отнести генерирование случайных чисел и неклонируемую идентификацию, которая, в свою очередь, является основой для методов и средств защиты цифровых устройств от нелегального использования и копирования. Реализованные ФНФ, как правило, представляют собой цифровые схемы, которые достаточно легко проектируются и характеризуются. Однако изготовление схемы ФНФ с заведомо известными характеристиками практически невозможно. Существует достаточное разнообразие схем цифровых ФНФ, большинство из которых основаны на уникальности задержек распространения сигналов. К подобным схемам можно отнести ФНФ типа арбитр [2], ФНФ кольцевых осцилляторов [3], ФНФ конфигурируемых кольцевых осцилляторов [4], ФНФ типа бабочка [5] и др. В настоящей работе делается попытка обобщения подобного класса ФНФ путем создания модели, основанной на множестве симметричных путей, для которых оцениваются значения задержек распространения сигналов. Показано, что наличие симметрии является определяющим фактором работоспособности схем ФНФ. Если симметрия путей может быть достигнута при проектировании и изготовлении заказных СБИС, то для технологий программируемых логических интегральных схем симметрия является практически недостижимой. На базе предложенной модели была спроектирована модифицированная схема ФНФ ККО, основанная на множестве конфигурируемых симметричных путей, с дальнейшей реализацией на программируемых логических интегральных схемах типа FPGA. Приводятся результаты моделирования и анализа основных параметров схем ККО и характеристик ФНФ ККО. Показано, что наличие заведомой асимметрии множества путей негативно сказывается на межкристальной уникальности ФНФ ККО.

ФНФ на основе симметричных путей. Большинство цифровых ФНФ основано на сравнении задержек распространения сигналов по путям, которые проектируются и изготавливаются исходя из предположения, что они являются идентичными. Под путем понимают часть цифровой схемы, для которой определен один вход и один выход. Эта часть схемы состоит из множества цифровых блоков и линий их межсоединений и обеспечивает задержку распространения цифрового сигнала от входа к выходу. Два пути являются симметричными, если множества блоков и межсоединений одного пути совпадают с аналогичными множествами другого пути, что служит необходимым условием равенства их задержек. Можно выделить следующие виды симметрии путей: проектную, топологическую и физическую. *Проектную симметрию* путей можно легко достичь на стадии проектирования цифрового устройства. В свою очередь, проектная симметрия может быть *функциональной* и *схемотехнической* (*структурной*). Так, функциональная симметрия двух путей обеспечивается заданием их эквивалентных поведенческих

проектных описаний, в то время как схемотехническая симметрия – путем задания эквивалентных структурных описаний. *Топологическая симметрия* может быть обеспечена идентичным расположением копий технологических элементов двух путей и геометрическим равенством их соответствующих межсоединений. *Физическая симметрия* может быть в пределе обеспечена при изготовлении полупроводникового кристалла. При этом физическая симметрия бывает двух типов: *внутрикристальная* и *межкристальная*. Если идеальную проектную и топологическую симметрию практически можно обеспечить, то идеальная физическая симметрия недостижима по многим причинам, в первую очередь связанным с девиациями материалов и технологических операций при изготовлении полупроводниковых кристаллов. Это выражается в разности задержек распространения сигналов по симметричным путям, а возможность регистрации такой разности лежит в основе построения цифровых ФНФ.

Для облегчения процесса создания симметричных путей и повышения их идентичности в схемах ФНФ часто применяют конфигурируемые симметричные пути [2, 4], для которых пара сравниваемых путей выбирается посредством задания значения запроса из конечного множества всех возможных запросов $CH \in \{CH_0, CH_1, CH_2, ..., CH_{C-1}\}$.

На рис. 1 приведены примеры симметричных и конфигурируемых симметричных путей, построенных на инверторах и трехстабильных буферах-усилителях. В обеих схемах присутствуют четыре симметричных пути. Для первой схемы (рис. 1, *a*) это пути *a*: (S_a, Q_a), *b*: (S_b, Q_b), *c*: (S_c, Q_c) и *d*: (S_d, Q_d). Во второй схеме каждый из C=4 путей конфигурируется одним из четырех значений $CH_n = (ch_n^0, ch_n^1)$, $n = \overline{0, C-1}$. Так, при $CH_0 = (0,0)$ конфигурируется путь p_0: (S, Q) от входа S до выхода Q через буферные элементы bt_0 и bt_1, а при $CH_3 = (1,1)$ – путь p_3: (S, Q) через элементы bt_2 и bt_3. Оба пути p_0: (S, Q) и p_3: (S, Q) по определению являются симметричными.



a) *b*)

Рис. 1. Схема симметричных путей (*a*) и конфигурируемых симметричных путей (*b*)
Fig. 1. Scheme of symmetrical paths (a) and configurable symmetrical paths (b)

Обе схемы на рис. 1 являются источниками задержек распространения сигналов от входа S до соответствующих выходов. Например, задержка для пути (S_a, Q_a) первой схемы представляет собой суммарную задержку распространения сигнала через два инвертора и транспортную задержку линий межсоединений, связывающих вход S_a с выходом Q_a.

При проектировании представленных схем по технологии заказных СБИС (ASIC) возможно обеспечение проектной и топологической симметрии всех путей. Однако физическая симметрия этих путей после реализации будет нарушена и приобретет уникальный характер в силу многих случайных факторов. С точки зрения обеспечения физической симметрии и экономии аппаратных ресурсов схемы с конфигурируемыми путями являются более предпочтительными. Примерами могут служить схемы ФНФ типа арбитр [2], ФНФ ККО [4] и др.

В общем случае задержку распространения сигнала по физическому пути можно выразить как $\Delta = \Delta_s + \delta_r$, где Δ_s является статической (проектной) составляющей задержки, а $\delta_r \ll \Delta_s$ – динамической случайной составляющей, значение и знак которой определяются внутри- и межкристальными вариациями в изготовленной схеме. При проектировании и параметрическом моделировании цифровых схем значение Δ_s фиксируется для выбранной технологии и условий эксплуатации (design corners) [6]. При этом случайная составляющая δ_r представляется тремя значениями: $\min(\delta_r)$, $\max(\delta_r)$ и $\text{typ}(\delta_r)$, где последнее значение является математическим ожиданием δ_r. Подобное упрощение в представлении задержек нашло применение в стандартном формате описания задержек SDF[1] (Standard Delay Format), широко используемом в современных САПР и средствах статического временного анализа STA[2] (Static Time Analysis).

При проектировании пар симметричных путей можно достичь равенства значений Δ_s, однако после физической реализации случайные значения δ_r приведут к нарушению заложенной на стадии проектирования симметрии.

Результатом сравнения задержек Δ^a и Δ^b двух симметричных путей *a* и *b*, выбранных по некоторому правилу в зависимости от значения запросов, является случайное значение при равенстве их статических составляющих:

$$\Delta^a - \Delta^b = \Delta_s^a - \Delta_s^b + \delta_r^a - \delta_r^b = \delta_r^a - \delta_r^b. \quad (1)$$

Знак приведенной разницы (1) определяет уникальность выбранных симметричных путей и ответ $R \in \{0,1\}$ по следующему правилу:

$$R = \begin{cases} 0, \text{ если } \delta_r^a < \delta_r^b, \\ 1, \text{ если } \delta_r^a \geq \delta_r^b. \end{cases} \quad (2)$$

Если пара путей заведомо является асимметричной, то на различных копиях схемы может наблюдаться одинаковое постоянное значение *R* для выбранного запроса.

Обобщенная модель ФНФ, основанная на сравнении задержек распространения сигналов по паре симметричных путей, изображена на рис. 2.

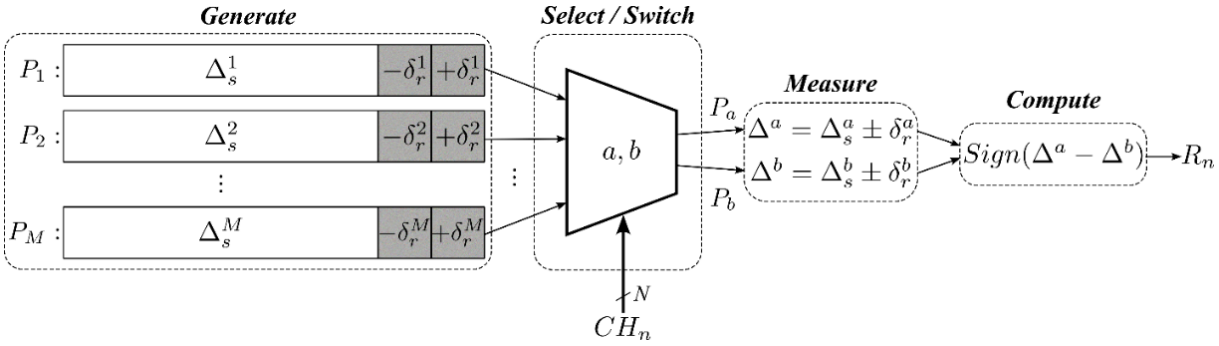


Рис. 2. Обобщенная модель ФНФ
Fig. 2. General PUF model

[1]IEEE Standard for Standard Delay Format (SDF) for the Electronic Design Process : IEEE Std 1497-2001. – 14 Dec. 2001. – 80 p. – DOI: 10.1109/IEEESTD.2001.93359.

[2]Mishagli, D. Statistical Static Timing Analysis of VLSI as the Statistics of Correlated Extremes / D. Mishagli, E. Koskin, E. Blokhina // Arxiv. – URL: https://arxiv.org/html/2401.03559v1 (date of access: 22.01.2025).

Представленная модель содержит четыре основные фазы, последовательность которых приводит к вычислению бита ответа R_n на поданное значение N-разрядного запроса CH_n:

Generate: генерирование множества симметричных путей $P_1, P_2, \ldots, P_M$ при физической реализации цифрового устройства, когда определяются характерные для них уникальные случайные компоненты δ_r^m, $m = \overline{1, M}$;

Select/Switch: при работе готового устройства по значению N-разрядного запроса CH_n ($N \le \lceil \log_2 C_M^2 \rceil$) определенным образом выбирается (*Select*) пара путей с индексами a и b ($a \neq b$), которая далее передается на следующую фазу в прямом $\{P_a, P_b\}$ либо обратном (*Switch*) $\{P_b, P_a\}$ порядке;

Measure: встроенными средствами цифрового устройства осуществляется измерение значений Δ^a и Δ^b для выбранной пары путей;

Compute: согласно равенству (1) осуществляется вычисление бинарного ответа R_n путем установления знака разницы измеренных значений Δ^a и Δ^b.

Общей фазой для всех типов ФНФ является *Generate*. Например, для ФНФ типа арбитр дополнительно выделяют фазы *Select* и *Switch* [7], при этом фаза *Measure* совмещена с фазой *Compute* и выполняется схемой арбитра. Для классической схемы ФНФ кольцевого осциллятора [3] характерны все фазы, представленные на рис. 2.

Характеристики ФНФ. С математической точки зрения ФНФ можно представить как сюръективное отображение $PUF : CH \to R$. В этом изображении каждому уникальному запросу CH_n, являющемуся элементом множества запросов $CH = \{CH_0, CH_1, ..., CH_{2^N-1}\}$, где $CH_n = (ch_n^0, ch_n^1, ..., ch_n^{N-1})$ есть двоичный N-разрядный вектор, $ch_n^i \in \{0,1\}$, $i = \overline{0, N-1}$, $n = \overline{0, 2^N - 1}$, однозначно соответствует бит ответа $R_n \in \{0,1\}$. Таким образом, произвольная ФНФ представляется множеством пар запрос-ответ $< CH, R >= \{(CH_0, R_0), (CH_1, R_1), (CH_2, R_2), ..., (CH_{2^N-1}, R_{2^N-1})\}$.

При многократном извлечении значения ответа R_n для одной и той же пары путей (фиксированного значения запроса CH_n) знак результирующей разницы (1) может изменяться. Связано это со многими факторами, которые могут влиять на значения δ_r^a и δ_r^b, например температурными изменениями кристалла, нестабильностью питающего напряжения, девиацией других неконтролируемых параметров элементов схемы. В таком случае пара запрос-ответ (CH_n, R_n) признается метастабильной. В противном случае, при устойчивом повторении ответа R_n, пара признается стабильной. *Стабильность (stability)* пары на E повторяющихся запросах можно оценить как вероятность появления нулевого либо единичного ответа:

$$STA_{CH_n}^E = 2 \cdot \left| 0,5 - \frac{1}{E} \cdot \sum_{e=0}^{E-1} R_n^e \right|, \tag{3}$$

где R_n^e – ответ, полученный при e-м повторении запроса CH_n.

Тогда стабильность всей ФНФ можно оценить следующей формулой:

$$STA = \frac{1}{2^N} \cdot \sum_{n=0}^{2^N-1} STA_{CH_n}^E . \tag{4}$$

Пусть $< CH, 0 >^E$ есть множество пар запрос-ответ со стабильным нулевым ответом, для которых $STA_{CH_n}^E = 1$. Тогда $< CH, 1 >^E$ – множество пар со стабильным единичным ответом,

а $< CH, X >^E$ – множество пар с метастабильным ответом, $STA^E_{CH_n} < 1$. Очевидно, что $< CH, R >^E = < CH, 0 >^E \cup < CH, 1 >^E \cup < CH, X >^E$, $|< CH, R >^E| = 2^N$.

Соотношение множеств стабильных и метастабильных пар определяет одну из важнейших характеристик ФНФ – *надежность (reliability)*:

$$REL = 1 - \frac{\left| < CR, X >^E \right|}{\left| < CR, 0 >^E \right| + \left| < CR, 1 >^E \right|}. \quad (5)$$

Множество стабильных пар может быть применено для решения задачи уникальной идентификации цифрового устройства, в то время как множество метастабильных пар – для генерирования случайных данных. В задачах уникальной идентификации метастабильные пары нивелируются, например, путем их стабилизации при помощи механизмов помехоустойчивого декодирования либо путем исключения их из рассмотрения. Последующие характеристики ФНФ основаны на утверждении, что $< CH, R >^E = < CH, 0 >^E \cup < CH, 1 >^E$.

Другой важной характеристикой ФНФ является *случайность*, определяемая соотношением мощностей множеств пар $< CH, 0 >$ и $< CH, 1 >$ и выражаемая нормированной метрикой *единообразия (uniformity)*:

$$UNI = 1 - 2 \cdot \left| 0,5 - \frac{|< CH, 1 >|}{|< CH, R >|} \right|. \quad (6)$$

Если мощность множества $|< CH, R >|$ достаточно велика, например N=64 и более, то для получения вышеописанных и других характеристик используют меньшее число запросов $T \ll 2^N$, равномерно распределенных по множеству всех возможных запросов и генерируемых, как правило, псевдослучайным образом.

Так, для сравнения двух копий i и j ФНФ применяют метрику *уникальности (uniqueness)*:

$$UNQ_{i,j} = 1 - \frac{1}{T} \cdot \left| < CH, R >_i \bigcap < CH, R >_j \right|, \quad (7)$$

по сути, представляющую собой удельное расстояние по Хэммингу между двумя векторами ответов двух ФНФ, полученных при подаче T различных запросов.

При сравнении k >2 копий ФНФ можно применять формулу

$$UNQ_k = \frac{1}{C_k^2} \cdot \sum_{i=1}^{k-1} \sum_{j=2}^{k} UNQ_{i,j}, \quad (8)$$

определяющую математическое ожидание уникальности всех возможных пар сравниваемых копий.

Метрика UNQ_k, рассчитанная для копий ФНФ, реализованных на одном кристалле, определяет *внутрикристальную уникальность* (*Intra* UNQ_k), а для копий ФНФ, реализованных на идентичных кристаллах, – *межкристальную уникальность* (*Inter* UNQ_k).

При реализации уникальной идентификации одной из важных задач исследователей и проектировщиков схем ФНФ является достижение значений перечисленных характеристик (4)–(7) близкими к значению 1, а характеристики (8) – к значению 0,5. В то же время при проектировании схем генераторов случайных чисел стремятся достичь значений *STA* и *REL*, близких к 0.

Обеспечение симметрии путей является определяющей задачей при построении схем ФНФ с высокими значениями их характеристик. В противном случае, если пары исследуемых путей не являются симметричными, то это в первую очередь негативно сказывается на такой характеристике, как уникальность. Как уже было отмечено, для ASIC-технологий возможно обеспечение симметрии конфигурируемых путей, однако для ПЛИС подобная реализация затруднена. С одной стороны, ПЛИС, например, типа FPGA, по сути, является полупроводниковым кристаллом, выполненным по технологии ASIC, но пути в ней реализуются посредством конфигурируемых связей, имеющих заведомую физическую асимметрию [8]. Кроме того, задержки распространения сигналов по конфигурируемым межсоединениям в FPGA заметно превосходят задержки на функциональных элементах, реализуемых, например, на LUT-блоках [9]. Рассмотрим эти особенности на примере реализации ФНФ типа ККО на FPGA и оценим приведенные основные характеристики данного вида ФНФ.

ФНФ типа ККО. Общая схема ККО (англ. CRO, Configurable Ring Oscillator) может быть представлена в качестве схемы задержки и управляемого инвертора, реализованного, как правило, при помощи вентиля NAND2, которые объединены петлей обратной связи [10]. Схема задержки при этом представляет собой M конфигурируемых симметричных путей (фаза *Generate*). Для построения схемы ФНФ на основе ККО необходимо наличие синхронных счетчиков CNT_1 и CNT_2 (фаза *Measure*), схемы компаратора *CMP* (фаза *Compute*) и устройства управления *CONTROL*, вырабатывающего необходимые последовательности сигналов. Кроме этого, *CONTROL* осуществляет выработку значений двух конфигураций CH_a и CH_b для ККО на основе подаваемого значения запроса CH_n (фаза *Select/Switch*) (рис. 3).

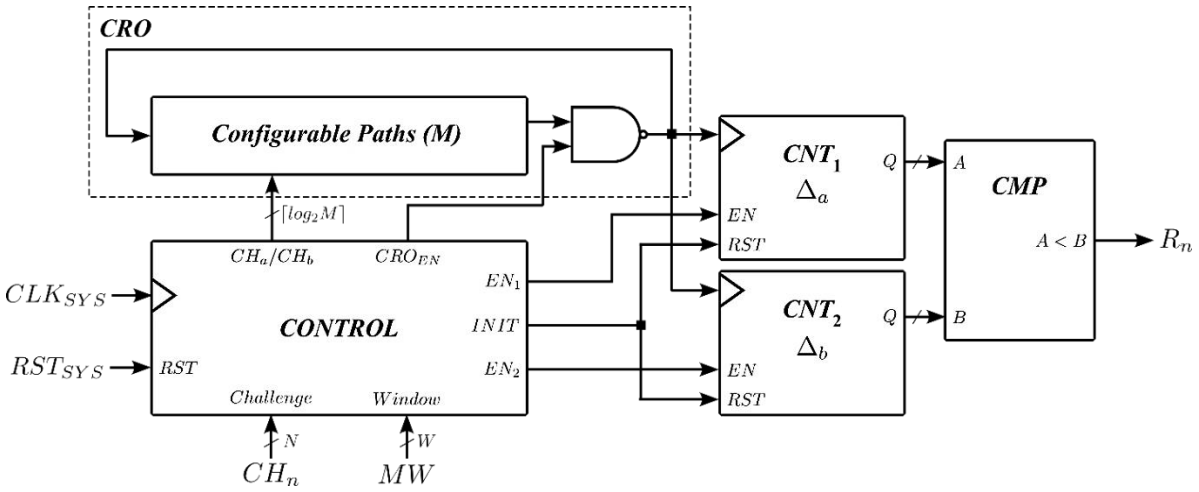


Рис. 3. Обобщенная модель ФНФ типа ККО

Fig. 3. Generalized model of CRO PUF

Изначально значения счетчиков обнуляются путем подачи сигнала *INIT* от блока управления. Затем на основе поданного запроса CH_n вырабатываются два значения конфигурации CH_a и CH_b, которые последовательно подаются на схему ККО. Выбранный путь по первичному значению CH_a из множества M симметричных путей формирует уникальную задержку в цепи обратной связи схемы ККО, которая оценивается на счетчике CNT_1 путем подсчета полных периодов генерируемого сигнала во временном окне измерения, задаваемого входным значением *MW* блока управления. Описанная последовательность действий повторяется для конфигурации CH_b с подсчетом полных периодов сигнала на счетчике CNT_2. В итоге значения двух счетчиков сравниваются на компараторе с выработкой бинарного ответа R_n. Рассмотренная схема может быть упрощена в части применения одного счетчика с прямым счетом для конфигурации CH_a и обратным для конфигурации CH_b в дополнительном коде [11]. Тогда итоговый знак значения счета и будет определять ответ схемы R_n.

Рассмотрим практическую схему ФНФ ККО, реализованную на ПЛИС типа FPGA Xilinx ZYNQ 7000, которая имеет следующие параметры: M=1024 и N=19. Схема конфигурируемых симметричных путей построена на пяти мультиплексорах MUX4x1. Все мультиплексоры последовательно соединены между собой и управляются 10-разрядной шиной конфигурации CF. Выбор такой схемы был обусловлен компактным размещением схем MUX4x1 на технологических элементах LUT6 кристалла FPGA. На рис. 4 показан результат технологического синтеза исследуемой схемы, включающий в себя пять блоков LUT6 ($L0$ – $L4$) и блок LUT2 ($A0$), который реализует управляемый инвертор схемы ККО.

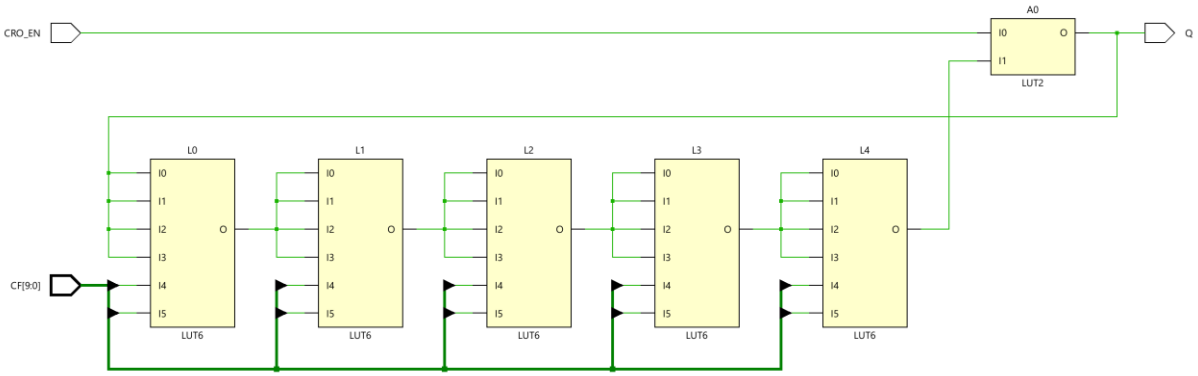
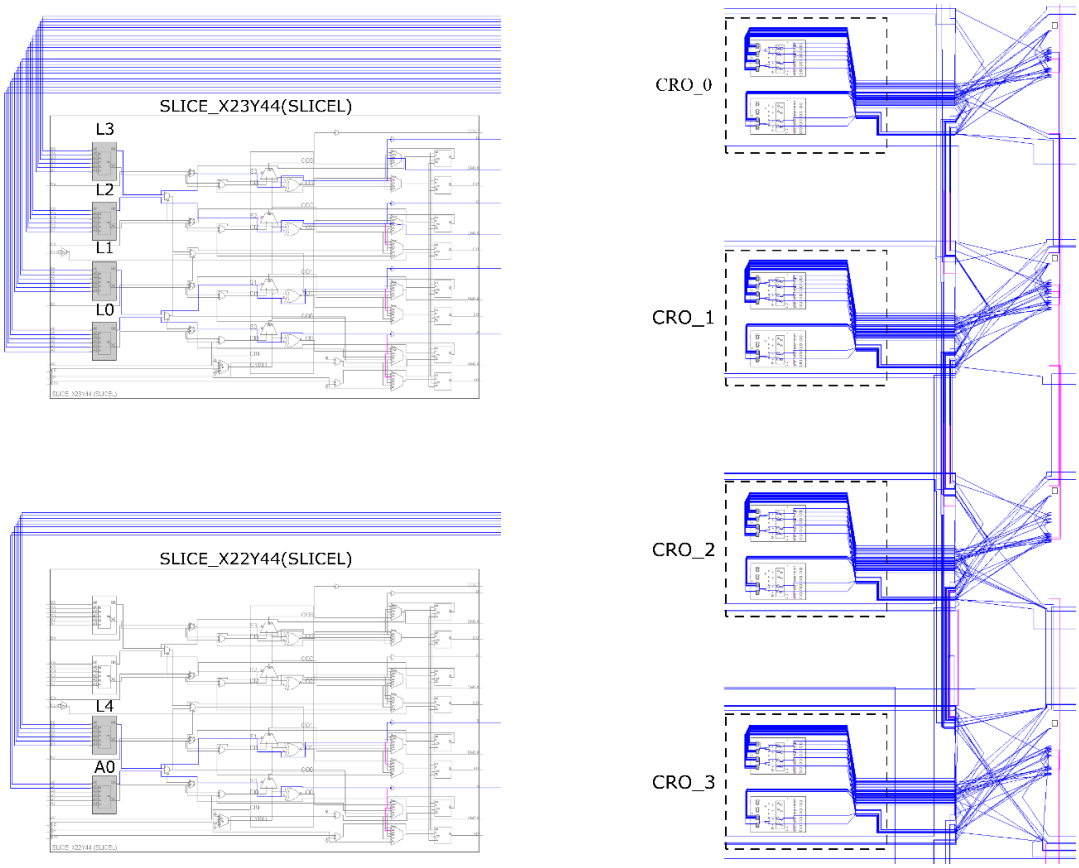


Рис. 4. Схема исследуемого ККО

Fig. 4. Scheme of studied CRO



a) *b)*

Рис. 5. Фиксированное размещение компонент одной схемы ККО (*a*) и четырех схем ККО (*b*) на ресурсах FPGA

Fig. 5. Fixed placement of components of one CRO circuit (a) and four CRO circuits (b) on FPGA resources

Для оценки основных характеристик ФНФ ККО в проекте были спроектированы четыре идентичные схемы (CRO_0 – CRO_3), впоследствии реализованные на четырех идентичных кристаллах ПЛИС ($FPGA_0$ – $FPGA_3$). Непосредственно перед реализацией были предварительно оценены симметричные пути посредством измерения периодов генерируемых сигналов при помощи параметрических моделей схем в различных сценариях моделирования, полученных после технологического синтеза. Кроме того, оценка симметрии производилась в двух вариантах размещения схем ККО: варианте автоматического размещения LUT-блоков и трассировки их межсоединений (Auto) и варианте фиксированного размещения (Fixed), при котором на этапе задания проектного описания указывалось конкретное размещение LUT-блоков, а межсоединения закреплялись путем описания их подключений к конкретным физическим входам. После этого фиксированное размещение одной схемы дублировалось на смежных ресурсах идентичных SLICE-блоков для обеспечения максимального подобия четырех компонент ФНФ ККО (рис. 5).

Результаты экспериментальных исследований. До реализации схем оценим симметрию их путей при помощи параметрических моделей, полученных на стадии технологического синтеза.

В САПР Vivado и симуляторе QuestaSim есть возможность оценить два типа задержек распространения сигналов: минимальную (Min) и максимальную (Max) – для двух условий эксплуатации (design corners): Fast и Slow. С учетом двух видов размещения (Auto и Fixed) можно получить восемь оценок задержек распространения сигналов для технологических моделей схем до их реализации. Например, задержка для блока LUT6 при условии Fixed/Fast/Min составляет 45 пс, при Fixed/Fast/Max – 56 пс, при Fixed/Slow/Min – 100 пс, а при Fixed/Slow/Max – 124 пс. Аналогичные оценки можно получить для межсоединений схемы. Например, значения задержек на линиях, соединяющих выход O блока L3 со входами I0, I1, I2 и I3 блока L4 (см. рис. 4), при условии Fixed/Fast/Min составляют: для линии L3/O:L4/I0 – 390,7 пс; для линии L3/O:L4/I1 – 340,4 пс; для линии L3/O:L4/I2 – 226 пс, а для линии L3/O:L4/I3 – 314,3 пс соответственно. На этом примере видно, что значения задержек на линиях межсоединений многократно превышают задержки на LUT-блоках и являются неравными. Это приводит к реализации заведомо асимметричных путей. Оценку периодов τ сигналов, генерируемых всеми четырьмя компонентами ККО на всех возможных 2^{10} конфигурациях, произведем на восьми моделях и двух вариантах их реализации (Auto и Fixed) на ПЛИС.

На рис. 6 и 7 приведены значения плотностей вероятностей моделируемых и реальных периодов, а в табл. 1 – основные статистические и вероятностные характеристики периодов для всех моделей и реальных схем ККО, реализованных на четырех ПЛИС типа FPGA. На рис. 6 видно, что реальные периоды сигналов схем ККО лежат ближе к модельным оценкам Fast/Max вне зависимости от типа размещения. Больший разброс (разнообразие) значений периодов наблюдается при модельных оценках автоматического размещения компонент ККО, для которых также характерны меньшие значения самих периодов, обусловленные более короткими межсоединениями структурных элементов в сравнении с их фиксированным размещением.

Схожие оценки получены и на реальных схемах (рис. 7, *a* и *b*), для которых также были оценены распределения периодов каждой схемы ККО при реализации на других ПЛИС. При этом наблюдается схожесть относительного распределения периодов на различных кристаллах вне зависимости от типа использованного размещения (рис. 7, *c* и *d*).

Если фиксированное размещение позволило сделать более схожими компоненты ККО, то различие в симметричных путях все равно осталось существенным. Так, для реальных схем диапазоны изменения периодов в автоматическом и фиксированном размещении составляют 3,295 и 3,803 нс соответственно (табл. 2).

Рассмотрим, как влияют приведенные данные на основные характеристики схем ФНФ ККО. Так, до реализации на ПЛИС на полученных параметрических моделях есть возможность оценить такие характеристики, как единообразие (6) и внутрикристальная уникальность (7). Для этого были созданы программные модели ФНФ на базе синтезированных схем ККО,

для которых получены $T=10^4$ ответов на псевдослучайно сгенерированные запросы. Аналогичные запросы были поданы и на реальные схемы ККО, в том числе для оценки других характеристик.

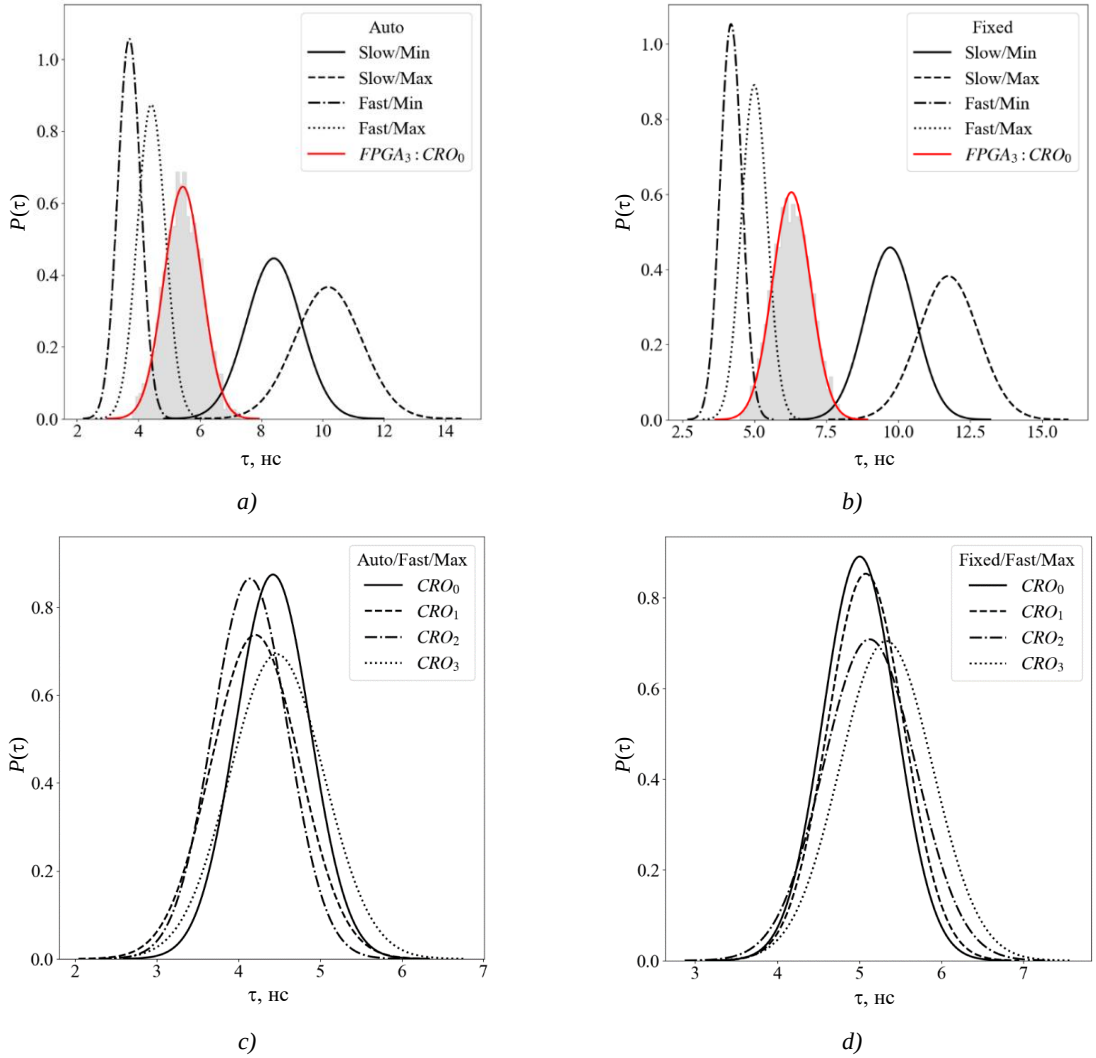


Рис. 6. Плотность вероятности $P(\tau)$ периодов τ компоненты CRO_0 в различных сценариях моделирования и типах размещения в сравнении с реальной компонентой на $FPGA_3$ (*a*, *b*) и для моделей всех компонент в сценарии Fast/Max (*c*, *d*)

Fig. 6. Probability density $P(\tau)$ of periods τ of component CRO_0 in various modeling scenarios and placement types compare to a real component on $FPGA_3$ (a, b) and for models of all components in the Fast/Max scenario (c, d)

В табл. 3 приведены результаты полученных характеристик *UNI*, а в табл. 4 – характеристик внутрикристальной уникальности UNQ_{ij} и UNQ_4 для всех моделей и схем ФНФ ККО. Из представленных данных видно, что все модели и схемы ФНФ ККО обладают достаточно высокими значениями метрики единообразия вне зависимости от типа размещения компонент на кристалле FPGA и в различных сценариях моделирования. Обусловлено это в первую очередь подавляющим большинством реализованных путей различной протяженности в схемах ККО, что асимптотически позволяет генерировать равные по мощности множества единичных и нулевых ответов (см. рис. 2).

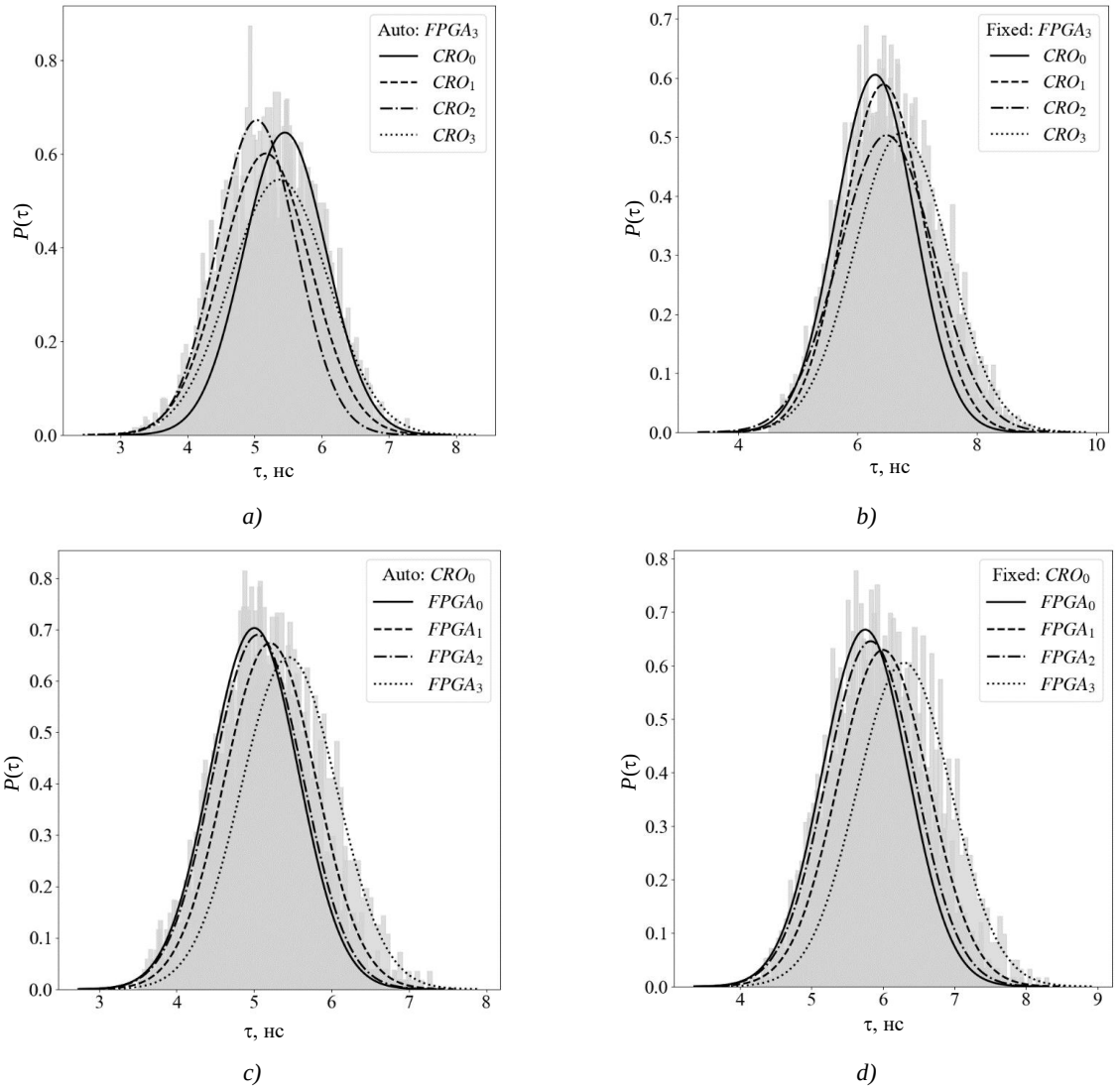


Рис. 7. Плотность вероятности $P(\tau)$ периодов τ всех компонент в различных типах размещения на кристалле $FPGA_3$ (*a, b*) и для компоненты CRO_0, реализованной на различных ПЛИС (*c, d*)

Fig. 7. Probability density $P(\tau)$ of periods τ of all components in various placement types on the $FPGA_3$ (a, b) and for the CRO_0 component implemented on various FPGAs (c, d)

В свою очередь, метрика внутрикристальной уникальности схем ФНФ ККО существенно снижается при фиксированном варианте размещения компонент в сравнении с их автоматическим размещением (см. табл. 4). Это может быть объяснено большой схожестью технологических схем ККО, которые после синтеза располагаются на идентичных ресурсах ПЛИС (см. рис. 5). Подобное поведение значений *Intra INQ* наблюдается как для моделей, так и для реальных схем ККО.

Еще меньшее различие между схемами ФНФ ККО наблюдается при реализации их копий на идентичных ПЛИС, о чем свидетельствуют крайне малые значения метрики *Inter UNQ* (см. табл. 5), которые практически не зависят от типа использованного размещения компонент схем.

Что касается метрики стабильности *STA*, то ее значения, так же как и значения единообразия для всех исследуемых схем ККО, реализованных на различных ПЛИС, являются весьма высокими (две-три девятки после запятой). Например, для схемы CRO_0, реализованной на $FPGA_0$ при автоматическом размещении, было получено значение *STA*=0,99905, а для этой же схемы

в фиксированном размещении – значение *STA*=0,99926. Вычисление данной метрики проходило при параметрах $T=10^4$ и $E=100$.

Таблица 1
Характеристики периодов сигналов моделей ККО в различных сценариях моделирования и типах размещения
Table 1
Characteristics of signal periods of CRO models in various simulation scenarios and placement types

Сценарий *Scenario*		ККО *CRO*	min(*T*), нс min(*T*), *ns*		max(*T*), нс max(*T*), *ns*		Математическое ожидание μ, нс *Expected value* μ, *ns*		СКО σ, нс *Standard deviation* σ, *ns*		σ/μ, %	
			Auto	Fixed	Auto	Fixed	Auto	Fixed	Auto	Fixed	Auto	Fixed
Slow	Max	CRO_0	7,212	8,876	13,436	14,918	10,185	11,748	1,088	1,043	10,685	8,882
		CRO_1	6,410	8,846	12,698	15,098	9,789	11,885	1,260	1,082	12,869	9,103
		CRO_2	6,480	8,846	12,434	15,902	9,701	12,028	1,102	1,255	11,354	10,433
		CRO_3	6,414	8,872	13,394	15,902	10,292	12,354	1,277	1,273	12,410	10,305
	Min	CRO_0	5,964	7,340	11,090	12,362	8,416	9,723	0,894	0,870	10,621	8,949
		CRO_1	5,138	7,310	10,232	12,514	7,972	9,845	1,033	0,904	12,952	9,185
		CRO_2	5,380	7,310	10,266	13,192	8,030	9,955	0,905	1,052	11,267	10,566
		CRO_3	5,310	7,336	11,056	13,186	8,524	10,235	1,058	1,063	12,412	10,388
Fast	Max	CRO_0	3,178	3,782	5,778	6,392	4,423	5,005	0,456	0,448	10,318	8,949
		CRO_1	2,764	3,752	5,458	6,448	4,207	5,077	0,541	0,468	12,870	9,212
		CRO_2	2,828	3,752	5,320	6,860	4,143	5,125	0,461	0,563	11,132	10,991
		CRO_3	2,750	3,778	5,842	6,874	4,471	5,327	0,576	0,566	12,876	10,632
	Min	CRO_0	2,662	3,168	4,824	5,362	3,702	4,195	0,377	0,379	10,191	9,030
		CRO_1	2,268	3,138	4,504	5,406	3,497	4,257	0,453	0,396	12,946	9,312
		CRO_2	2,388	3,138	4,446	5,764	3,478	4,293	0,382	0,481	10,986	11,198
		CRO_3	2,310	3,164	4,882	5,776	3,755	4,472	0,485	0,481	12,918	10,751

Таблица 2
Характеристики периодов сигналов реальных ККО в различных типах размещения
на различных ПЛИС
Table 2
Characteristics of signal periods of real CROs in various types of placement on different FPGAs

ПЛИС *FPGA*	ККО *CRO*	min(*T*), нс min(*T*), *ns*		max(*T*), нс max(*T*), *ns*		Математическое ожидание μ, нс *Expected value* μ, *ns*		СКО σ, нс *Standard deviation* σ, *ns*		σ/μ, %	
		Auto	Fixed	Auto	Fixed	Auto	Fixed	Auto	Fixed	Auto	Fixed
$FPGA_0$	CRO_0	3,489	4,133	6,713	7,590	5,003	5,752	0,567	0,598	11,343	10,392
	CRO_1	3,124	4,111	6,092	7,641	4,718	5,834	0,603	0,612	12,773	10,485
	CRO_2	3,062	4,136	5,988	8,105	4,594	5,911	0,539	0,718	11,734	12,142
	CRO_3	2,922	4,137	6,544	8,055	4,939	6,094	0,671	0,705	13,590	11,564
$FPGA_1$	CRO_0	3,627	4,296	6,989	7,968	5,209	6,001	0,592	0,634	11,368	10,561
	CRO_1	3,254	4,307	6,348	8,038	4,932	6,138	0,631	0,650	12,798	10,594
	CRO_2	3,187	4,334	6,217	8,548	4,796	6,205	0,560	0,762	11,671	12,276
	CRO_3	3,060	4,344	6,802	8,468	5,149	6,404	0,702	0,742	13,625	11,584
$FPGA_2$	CRO_0	3,506	4,157	6,779	7,732	5,048	5,827	0,578	0,618	11,457	10,602
	CRO_1	3,151	4,161	6,170	7,810	4,780	5,964	0,616	0,633	12,885	10,607
	CRO_2	3,070	4,171	6,011	8,203	4,632	5,973	0,542	0,734	11,693	12,292
	CRO_3	2,985	4,202	6,605	8,192	5,006	6,196	0,675	0,720	13,491	11,622
$FPGA_3$	CRO_0	3,796	4,504	7,295	8,316	5,450	6,294	0,618	0,659	11,333	10,467
	CRO_1	3,407	4,522	6,681	8,379	5,168	6,435	0,664	0,677	12,840	10,524
	CRO_2	3,294	4,514	6,513	8,893	5,031	6,486	0,593	0,793	11,790	12,225
	CRO_3	3,180	4,547	7,093	8,925	5,367	6,731	0,731	0,788	13,628	11,700

Таблица 3
Значения метрики единообразия *UNI* схем ФНФ ККО в различных сценариях моделирования и размещения

Table 3
Values of the UNI uniformity metric of CRO PUF schemes in various modeling and placement scenarios

Сценарий/ ПЛИС *Scenario/ FPGA*	CRO_0		CRO_1		CRO_2		CRO_3	
	Auto	Fixed	Auto	Fixed	Auto	Fixed	Auto	Fixed
Slow/Min	0,9914	0,9894	0,993	0,9972	0,9934	0,992	0,9992	0,9894
Slow/Max	0,9956	0,9898	0,9932	0,9952	0,9914	0,9914	0,9984	0,99
Fast/Min	0,993	0,988	0,9976	0,9968	0,993	0,9932	0,997	0,993
Fast/Max	0,9958	0,99	0,9932	0,9968	0,9928	0,994	0,9972	0,9928
$FPGA_0$	0,9928	0,994	0,9908	0,9996	0,9914	0,994	0,9998	0,9966
$FPGA_1$	0,9928	0,9912	0,9908	0,9968	0,99	0,9912	0,9996	0,9966
$FPGA_2$	0,993	0,9942	0,9918	0,9984	0,993	0,9936	0,9992	0,9988
$FPGA_3$	0,9938	0,9914	0,9916	0,9966	0,9888	0,9952	0,9994	0,9982

Таблица 4
Значения метрики внутрикристальной уникальности схем ФНФ ККО

Table 4
Values of the metric of intra-chip uniqueness of CRO PUF circuits

Сценарий/ ПЛИС *Scenario/ FPGA*			Внутрикристальная уникальность *Intra UNQ* *Intra-chip uniqueness Intra UNQ*						
			$UNQ_{0,1}$	$UNQ_{0,2}$	$UNQ_{0,3}$	$UNQ_{1,2}$	$UNQ_{1,3}$	$UNQ_{2,3}$	UNQ_4
Auto	Slow	Min	0,4966	0,2578	0,4893	0,4194	0,3091	0,3823	0,3924
		Max	0,4836	0,2539	0,4842	0,3949	0,313	0,3783	0,3846
	Fast	Min	0,5061	0,2568	0,4942	0,4107	0,3327	0,394	0,3991
		Max	0,4899	0,2547	0,4869	0,389	0,33	0,3854	0,3893
Fixed	Slow	Min	0,1447	0,0779	0,1106	0,1928	0,0949	0,1545	0,1292
		Max	0,1413	0,0764	0,1099	0,1889	0,093	0,1517	0,1269
	Fast	Min	0,1412	0,0956	0,1171	0,1926	0,0857	0,1719	0,134
		Max	0,1386	0,0934	0,1176	0,189	0,084	0,1676	0,1317
Auto	$FPGA_0$		0,4798	0,4798	0,4719	0,3947	0,3381	0,3856	0,3882
	$FPGA_1$		0,4854	0,4854	0,472	0,4062	0,3422	0,3858	0,3918
	$FPGA_2$		0,4832	0,4832	0,4715	0,403	0,3423	0,3839	0,3909
	$FPGA_3$		0,4843	0,4843	0,475	0,4038	0,3421	0,3853	0,3922
Fixed	$FPGA_0$		0,1214	0,1214	0,0965	0,161	0,0697	0,1463	0,1124
	$FPGA_1$		0,1274	0,1274	0,1001	0,1658	0,0753	0,1463	0,1145
	$FPGA_2$		0,1255	0,1255	0,1005	0,1712	0,0732	0,148	0,1161
	$FPGA_3$		0,1214	0,1214	0,0968	0,1665	0,0716	0,1501	0,1144

На рис. 8 изображены графики распределения значений $STA^E_{CH_n}$ по всем копиям схем ФНФ ККО, реализованных на ПЛИС $FPGA_3$ для двух вариантов размещения (по осям абсцисс приведены условные индексы использованных псевдослучайных запросов). Видно, что большее число нестабильных пар запрос-ответ наблюдается для фиксированного размещения в сравнении с автоматическим размещением. Это, в свою очередь, сказывается на значениях метрик стабильности *STA* и надежности *REL*. Так, среднее значение метрики *STA* для всех схем при автоматическом размещении составляет 0,9993, а при фиксированном – 0,9989.

Таблица 5
Значения метрик межкристальной уникальности схем ФНФ ККО

Table 5
Values of the metric of inter-chip uniqueness of CRO PUF circuits

Размещение *Placement*	ККО *CRO*	Межкристальная уникальность *Inter UNQ* *Inter-chip uniqueness Inter UNQ*						
		$UNQ_{0,1}$	$UNQ_{0,2}$	$UNQ_{0,3}$	$UNQ_{1,2}$	$UNQ_{1,3}$	$UNQ_{2,3}$	UNQ_4
Auto	CRO_0	0,01	0,0109	0,0119	0,0103	0,0111	0,0096	0,0106
	CRO_1	0,0086	0,0051	0,0072	0,0079	0,0086	0,0067	0,0074
	CRO_2	0,0113	0,0094	0,0145	0,0079	0,0084	0,0099	0,0102
	CRO_3	0,0067	0,0055	0,0076	0,005	0,0055	0,0061	0,0061
Fixed	CRO_0	0,011	0,0089	0,0101	0,0093	0,0075	0,0106	0,0096
	CRO_1	0,0112	0,0154	0,0207	0,0098	0,0137	0,0161	0,0145
	CRO_2	0,0102	0,0108	0,0088	0,0104	0,0122	0,011	0,0106
	CRO_3	0,008	0,0101	0,0068	0,0105	0,0084	0,0067	0,0084

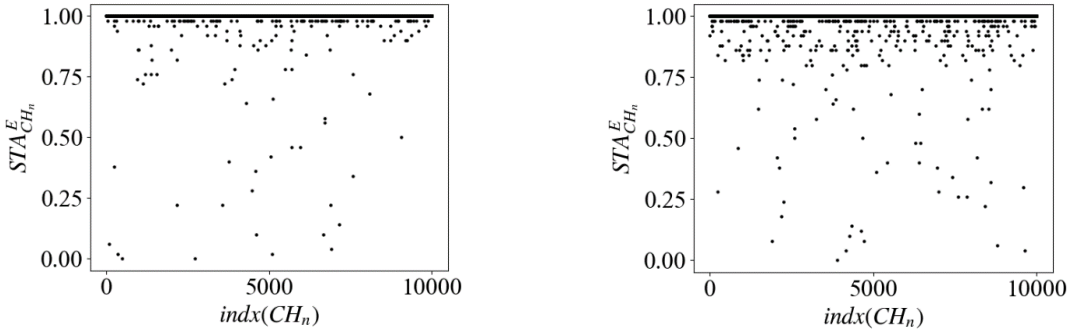


Рис. 8. Значения метрики $STA^E_{CH_n}$ для всех схем ФНФ ККО, реализованных в автоматическом (*a*) и фиксированном (*b*) расположении на $FPGA_3$

Fig. 8. $STA^E_{CH_n}$ metric values for all CRO PUF schemes implemented in automatic (a) and fixed (b) placement of $FPGA_3$

В табл. 6 представлены значения числа метастабильных пар запрос-ответ $|< CR, X >^E|$ (5) для всех исследуемых схем ФНФ ККО в двух сценариях размещения. Как видно из представленных данных, суммарное число метастабильных пар ФНФ при фиксированной реализации почти в 1,85 раза больше, чем при автоматическом размещении, что свидетельствует о большей схожести сравниваемых пар путей схем ККО.

Таблица 6
Число метастабильных пар ФНФ ККО

Table 6
The number of metastable pairs of CRO PUF

Размещение *Placement*	ПЛИС *FPGA*	$\|< CR, X >^E\|$			
		CRO_0	CRO_1	CRO_2	CRO_3
Auto	$FPGA_0$	86	45	14	33
	$FPGA_1$	79	28	24	29
	$FPGA_2$	59	27	29	29
	$FPGA_3$	81	23	28	37
Fixed	$FPGA_0$	89	40	103	89
	$FPGA_1$	77	27	118	32
	$FPGA_2$	74	24	139	46
	$FPGA_3$	78	42	145	84

Значение метрики надежности ФНФ *REL* лежит в диапазоне [0,9853; 0,9986], где наименьшее значение относится к схеме CRO_2, реализованной на $FPGA_3$ с фиксированным размещением, а наибольшее значение – к той же схеме, реализованной на $FPGA_0$ с автоматическим размещением.

Таким образом, можно констатировать, что реализованные на ПЛИС схемы ФНФ ККО обладают приемлемыми показателями единообразия, внутрикристальной уникальности, стабильности и надежности при автоматическом размещении их компонент.

Фиксированное размещение увеличивает степень схожести сравниваемых путей схем ККО, о чем свидетельствует уменьшение показателей внутрикристальной уникальности, стабильности и надежности. Однако вне зависимости от типа используемого размещения все схемы ФНФ ККО обладают неприемлемыми показателями межкристальной уникальности, что может являться сильным ограничением на их практическое применение для схем уникальной идентификации.

Заключение. В статье рассмотрены общие принципы проектирования ФНФ на основе конфигурируемых симметричных путей. Выделены четыре основные фазы функционирования подобных схем: *Generate, Select/Switch, Measure* и *Compute*. На примере схем ККО показано, что фаза *Generate* является определяющей для основных характеристик ФНФ при их реализации на ПЛИС. Высокая степень асимметрии конфигурируемых путей ККО, реализованных на программируемых межсоединениях, негативно сказывается на такой характеристике, как межкристальная уникальность. Было показано также, что некоторые характеристики ФНФ (единообразие и внутрикристальная уникальность) могут быть предварительно оценены на параметрических моделях после синтеза в зависимости от различных типов размещения компонент на кристалле. Проведенные эксперименты с различными схемами ФНФ ККО показали адекватность модельных оценок в сравнении с оценками, полученными на реальных ПЛИС.

Дальнейшие исследования могут быть направлены на поиск новых схемотехнических и аналитических методов улучшения характеристик уникальности для ФНФ, основанных на анализе задержек распространения сигналов по симметричным путям.

Список использованных источников

1. Secure System Design and Trustable Computing / ed.: Ch. H. Chang, M. Potkonjak. – Switzerland : Springer, 2016. – 549 p. – DOI: 10.1007/978-3-319-14971-4.

2. Hemavathy, S. Arbiter PUF – a review of design, composition, and security aspects / S. Hemavathy, V. S. K. Bhaaskaran // IEEE Access. – 2023. – Vol. 11. – P. 33979–34004. – DOI: 10.1109/ACCESS.2023.3264016.

3. Maiti, A. Improved ring oscillator PUF: An FPGA-friendly secure primitive / A. Maiti, P. Schaumont // Journal of Cryptology. – 2011. – Vol. 24. – P. 375–397. – DOI: 10.1007/s00145-010-9088-4.

4. Configurable ring oscillator PUF using hybrid logic gates / D. Deng, S. Hou, Z. Wang, Y. Guo // IEEE Access. – 2020. – Vol. 8. – P. 161427–161437. – DOI: 10.1109/ACCESS.2020.3021205.

5. Extended abstract: The butterfly PUF protecting IP on every FPGA / S. S. Kumar, J. Guajardo, R. Maes [et al.] // Proc. of the IEEE Intern. Workshop on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 09 June 2008. – Anaheim, 2008. – P. 67–70. – DOI: 10.1109/HST.2008.4559053.

6. Corner models: Inaccurate at best, and it only gets worst / C. C. McAndrew, I.-S. Lim, B. Braswell, D. Garrity // Proc. of the IEEE 2013 Custom Integrated Circuits Conf., San Jose, CA, USA, 22–25 Sept. 2013. – San Jose, 2013. – P. 1–4. – DOI: 10.1109/CICC.2013.6658428.

7. Ярмолик, В. Н. Двухмерные физически неклонируемые функции типа арбитр / В. Н. Ярмолик, А. А. Иванюк // Информатика. – 2023. – Т. 20, № 1. – С. 7–26. – DOI: 10.37661/1816-0301-2023-20-1-7-26.

8. Global interconnections in FPGAs: modeling and performance analysis / T. Mak, C. D'Alessandro, P. Sedcole [et al.] // Proc. of Intern. Workshop on System Level Interconnect Prediction, Newcastle, United Kingdom, 05–08 April 2008. – Newcastle, 2008. – P. 51–58. – DOI: 10.1145/1353610.1353621.

9. Karnik, T. An empirical model for accurate estimation of routing delay in FPGAs / T. Karnik, S.-M. Kang // Proc. of IEEE Intern. Conf. on Computer Aided Design (ICCAD), San Jose, CA, USA, 05–09 Nov. 1995. – San Jose, 1995. – P. 328–331. – DOI: 10.1109/ICCAD.1995.480136.

10. Иванюк, А. А. Конфигурируемый кольцевой осциллятор с управляемыми межсоединениями / А. А. Иванюк, В. Н. Ярмолик // Безопасность информационных технологий. – 2024. – Т. 31, № 2. – С. 121–133. – DOI: 10.26583/bit.2024.2.08.

11. Иванюк, А. А. Физически неклонируемые функции на базе управляемого кольцевого осциллятора / А. А. Иванюк, В. Н. Ярмолик // Безопасность информационных технологий. – 2023. – Т. 30, № 3. – С. 90–103. – DOI: 10.26583/bit.2023.3.06.

References

1. Chang Ch. H., Potkonjak M. (eds.). *Secure System Design and Trustable Computing*. Switzerland, Springer, 2016, 549 p. DOI: 10.1007/978-3-319-14971-4.

2. Hemavathy S., Bhaaskaran V. S. K. Arbiter PUF – a review of design, composition, and security aspects. *IEEE Access*, 2023, vol. 11, pp. 33979–34004. DOI: 10.1109/ACCESS.2023.3264016.

3. Maiti A., Schaumont P. Improved ring oscillator PUF: An FPGA-friendly secure primitive. *Journal of Cryptology*, 2011, vol. 24, pp. 375–397. DOI: 10.1007/s00145-010-9088-4.

4. Deng D., Hou S., Wang Z., Guo Y. Configurable ring oscillator PUF using hybrid logic gates. *IEEE Access*, 2020, vol. 8, pp. 161427–161437. DOI: 10.1109/ACCESS.2020.3021205.

5. Kumar S. S., Guajardo J., Maes R., Schrijen G.-J., Tuyls P. Extended abstract: The butterfly PUF protecting IP on every FPGA. *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 09 June 2008*. Anaheim, 2008, pp. 67–70. DOI: 10.1109/HST.2008.4559053.

6. McAndrew C. C., Lim I.-S., Braswell B., Garrity D. Corner models: Inaccurate at best, and it only gets worst. *Proceedings of the IEEE 2013 Custom Integrated Circuits Conference, San Jose, CA, USA, 22–25 September 2013*. San Jose, 2013, pp. 1–4. DOI: 10.1109/CICC.2013.6658428.

7. Yarmolik V. N., Ivaniuk A. A. *2D physically unclonable functions of the arbiter type*. Informatika [*Informatics*], 2023, vol. 20, no. 1, pp. 7−26 (In Russ.). DOI: 10.37661/1816-0301-2023-20-1-7-26.

8. Mak T., D'Alessandro C., Sedcole P., Cheung P. Y. K., …, Luk W. Global interconnections in FPGAs: modeling and performance analysis. *Proceedings of International Workshop on System Level Interconnect Prediction, Newcastle, United Kingdom, 05–08 April 2008*. Newcastle, 2008, pp. 51–58. DOI: 10.1145/1353610.1353621.

9. Karnik T., Kang S.-M. An empirical model for accurate estimation of routing delay in FPGAs. *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD), San Jose, CA, USA, 05–09 November 1995*. San Jose, 1995, pp. 328–331. DOI: 10.1109/ICCAD.1995.480136.

10. Ivaniuk A. A., Yarmolik V. N. *Configurable ring oscillator with controlled interc*onnections. Bezopasnost` informatsionnykh tekhnologiy [*IT Security (Russia)*], 2024, vol. 31, no. 2, pp. 121–133 (In Russ.). DOI: 10.26583/bit.2024.2.08.

11. Ivaniuk A. A., Yarmolik V. N. *Physically unclonable functions based on a controlled ring oscillator*. [*IT Security (Russia)*], 2023, vol. 30, no. 3, pp. 90–103 (In Russ.). DOI: 10.26583/bit.2023.3.06.

Информация об авторе

Иванюк Александр Александрович, доктор технических наук, профессор, профессор кафедры информатики, Белорусский государственный университет информатики и радиоэлектроники.
E-mail: ivaniuk@bsuir.by

Information about the author

Alexander A. Ivaniuk, D. Sc. (Eng.), Prof., Prof. of Computer Science Department, Belarusian State University of Informatics and Radioelectronics.
E-mail: ivaniuk@bsuir.by

УДК 004.832, 519.6
DOI: 10.37661/1816-0301-2025-22-1-90-97

Оригинальная статья
Original Article

Адаптация модулярной системы счисления в пороговых схемах разделения секрета

А. Ф. Чернявский, Е. И. Козлова✉, В. С. Садов, А. А. Коляда

Белорусский государственный университет,
пр. Независимости, 4, Минск, 220030, Беларусь
✉E-mail: kozlova@bsu.by

Аннотация

Цели. Целью проведенных исследований является проверка применимости варианта адаптации модулярной системы счисления с использованием маскирующего преобразования с псевдослучайной целочисленной величиной к секрету-оригиналу S в модификации (k, n)-пороговой схемы Ади Шамира разделения секрета для сведения к теоретическому минимуму сложности расчета базовой интегральной характеристики.

Методы. Рассмотрена модификация схемы Ади Шамира разделения секрета в пороговой криптосистеме на основе модулярной арифметики (МА-криптосистеме) с генерацией долей участников разделения секрета в два этапа. Схема Шамира выбрана как оптимальная по параметрам сложности, ресурсоемкости, совершенности и идеальности. Кроме того, она масштабируема – количество участников можно увеличивать до порядка поля p, при этом не меняется способность восстанавливать секрет. Применено маскирующее преобразование с использованием слагаемого с псевдослучайной целочисленной величиной C для разделяемого секрета S и согласование диапазона изменения псевдослучайного параметра C и области изменения значений оригинала сигнала. Применена также интервально-модулярная форма числа значения секрета.

Результаты. Показано, что использование интервально-модулярной формы числа $\overline{S}$ – маскирующего преобразования с псевдослучайным параметром числа S секрета-оригинала – снижает сложность расчета базовых интервально-индексных характеристик при решении задач порогового кодирования практически до теоретического минимума. Адаптивное согласование диапазона изменений псевдослучайного параметра маскирующей функции с областью ее значений позволяет реализовать минимально избыточную модулярную декомпозицию функции маскирования при любом допустимом базисе оснований схемы.

Заключение. Результаты представленной работы позволяют для модулярных пороговых криптосистем разделения секрета в распределенных системах обработки данных сделать вывод о том, что применение линейной маскирующей функции и сужение области изменения маскирующего аналога $\tilde{S}$ секрета-оригинала S, допускающее при выбранных $p_1, p_2, \ldots, p_n$ минимально избыточное кодирование, обусловливают существенное снижение вычислительной сложности расчетных соотношений минимально-избыточной модулярной арифметики интегральных характеристик в рамках исследуемой модели. Благодаря этому достигается более высокий уровень производительности на стадии декодирования секрета-оригинала по сравнению с другими решениями.

Ключевые слова: минимальная избыточность, модулярное кодирование, маскирующее преобразование, пороговая криптосистема, секрет-оригинал, интервальные характеристики

Благодарности. Работа выполнена в рамках Государственной программы научных исследований «Цифровые и космические технологии, безопасность общества и государства» (подпрограмма «Цифровые технологии и космическая информатика», задание 5.1.6.3), 2021–2025 гг.

Для цитирования. Адаптация модулярной системы счисления в пороговых схемах разделения секрета / А. Ф. Чернявский, Е. И. Козлова, В. С. Садов, А. А. Коляда // Информатика. – 2025. − Т. 22, № 1. – С. 90–97. – DOI: 10.37661/1816-0301-2025-22-1-90-97.

Конфликт интересов. Авторы заявляют об отсутствии конфликта интересов.

Поступила в редакцию | Received 31.01.2025
Подписана в печать | Accepted 14.02.2025
Опубликована | Published 31.03.2025

Adaptation of the modular number system in threshold secret sharing schemes

Alexander F. Chernyavskiy, Alena I. Kazlova✉, Vasiliy S. Sadov, Andrei A. Kolyada

Belarusian State University,
av. Nezavisimosti, 4, Minsk, 220030, Belarus
✉*E-mail: kozlova@bsu.by*

Abstract

Objectives. The purpose of the research is to test the applicability of the adaptation of the modular number system using a masking transformation with a pseudo-random integer value to the original secret *S* in a modification of Adi Shamir's (*k*, *n*)-threshold secret sharing scheme to reduce the complexity of calculating the basic integral characteristic to a theoretical minimum.

Methods. A modification of Adi Shamir's secret sharing scheme in a threshold cryptosystem based on modular arithmetic (MA cryptosystem) with the generation of shares of secret sharing participants in two stages is considered. Shamir's scheme was chosen as optimal in terms of complexity, resource intensity, perfection and ideality; in addition, it is scalable – the number of participants can be increased to the order of the field *p*, without changing the ability to recover the secret. A masking transformation using a term with a pseudo-random integer value *C* for the shared secret *S*, the range of change of the pseudo-random parameter *C* agreed upon the range of changes in the values of the original signal is applied. The interval-modular form of the number of the secret value is applied too.

Results. It is shown that the use of the interval-modular form of the number $\bar{S}$ – a masking transformation with a pseudo-random parameter of the number *S* of the original secret – reduces the complexity of calculating basic interval-index characteristics when solving threshold coding problems almost to a theoretical minimum. Adaptive coordination of changes in the pseudo-random parameter of the masking function with the domain of its results makes it possible to implement a minimally redundant modular decomposition of the masking function for any admissible basis of the based scheme.

Conclusion. The results of the presented work allow to conclude for modular threshold cryptosystems of secret sharing in distributed data processing systems that the use of a linear masking function and narrowing the range of changes in the masking analogue $\tilde{S}$ of the original secret *S*, allowing for minimally redundant coding for the selected $p_1, p_2, \ldots, p_n$, causes a significant reduction in the computational complexity of the calculated minimal-redundant modular arithmetic relations of integral characteristics within the framework of the model under study. Due to which a higher level of performance is achieved at the stage of decoding the original secret compared to other solutions.

Keywords: minimal redundancy, modular coding, masking transformation, threshold cryptosystem, secret-original, interval characteristics

Acknowledgments. The work was carried out within the framework of the state scientific research program "Digital and space technologies, security of society and the state" (subprogram "Digital technologies and space informatics", task 5.1.6.3), 2021–2025.

For citation. Chernyavskiy A. F., Kazlova A. I., Sadov V. S., Kolyada A. A. *Adaptation of the modular number system in threshold secret sharing schemes*. Informatika [*Informatics*], 2025, vol. 22, no. 1, pp. 90–97 (In Russ.). DOI: 10.37661/1816-0301-2025-22-1-90-97.

Conflict of interest. The authors declare of no conflict of interest.

Введение. Обеспечение информационной безопасности информационных и инфокоммуникационных систем является сегодня одной из приоритетных задач как при разработке и реализации таких систем, так и при их эксплуатации. В процессе функционирования информационных систем применяются технологии активной безопасности, включающие такие методы защиты от прямых угроз, как периодическое обновление секретной информации и ее пространственное разделение. Одновременное использование этих методов позволяет повысить уровень безопасности систем криптографической защиты информации. Методы пространственного разделения информации основаны на возможности ее разделения между несколькими пользователями системы так, что применить ключ можно только с помощью части информации определенного числа участников разделения ключа из всего числа «хранителей» этих частей. Периодичность смены ключевой информации (в частности, криптографических ключей) в сочетании со случайным характером изменений и времени их проведения характерна для всех систем активной безопасности. Для того чтобы обеспечить доступность, конфиденциальность и целостность данных, в частности надежное хранение, применяются системы криптографического разделения секрета, отвечающие в том числе и требованиям периодичности, случайности и внезапности изменения ключей. К наиболее часто используемым системам такого типа относятся, например, пороговые схемы Шамира, Блэкли, Карнина – Грина – Хеллмана, Асмута – Блума и схема разделения секрета, основанная на эллиптической кривой [1–4]. Эти схемы решают задачу генерации ключей, применяя криптографически стойкие алгоритмы с использованием генератора псевдослучайной последовательности для получения таких последовательностей, которые статистически неотличимы от абсолютно случайных последовательностей, т. е. значения сгенерированной последовательности оказываются непредсказуемыми. Пороговая схема Шамира разделения секрета многими авторами указывается как наиболее эффективная среди известных. Хотя она не является наиболее быстрой, но оказывается оптимальной по ресурсоемкости, совершенности и идеальности и имеет простой способ разделения секрета [1–4].

Известно, что компьютерно-арифметической базой средств защиты информации является арифметика больших целых чисел [4]. При этом на практике эффективность вычислительного аппарата криптопреобразований определяется возможностями перевода вычислений из категории больших целых чисел в категорию целых чисел стандартной разрядности. Это обусловливает актуальность применения арифметики модулярных систем счисления (МСС) [4] для построения пороговых криптосхем разделения секрета. МСС обладает естественным кодовым параллелизмом. Вопрос производительности, в частности скорости проведения вычислений, является очень чувствительным в криптографических приложениях, что также обусловливает целесообразность применения в них модулярной арифметики. Модулярное кодирование позволяет достаточно просто произвести разделение секрета на части для участников с минимальными временными и аппаратными затратами в диапазонах больших чисел. Кроме задачи повышения производительности, применение МСС позволяет решить также задачу снижения трудоемкости процедуры восстановления ключа-оригинала по частичным секретам, выданным некоторой группе участников при его разделении.

В процессе анализа функциональных возможностей различных технологий вычисления установлено, что при использовании МСС введение в модулярный код минимальной избыточности существенно упрощает оценку интегральных характеристик и связанных с ними форм представления целых чисел [5, 6]. Это позволяет в том числе сократить до минимума временные и аппаратные затраты на выполнение операции восстановления ключа-оригинала. Интегральные характеристики модулярного кода (ИХМК) включают: ранговое число, интервальный индекс и его евклидовы компоненты.

Решающее правило, реализуемое (t, n)-пороговой системой разделения секрета, рассчитано на полное число $<n>$ и пороговое число $<t>$ абонентов. Ряд участников t разделения секрета из полного их числа n, которые могут получать секретные данные, считаются разрешенным множеством, или разрешенной коалицией участников. При этом в распределенных системах наиболее перспективной технологией защиты данных считается технология активной безопасности,

т. е. число абонентов такого множества должно быть больше одного. Множества участников, которые не могут получать секретные данные, относятся к запрещенной коалиции (множеству).

Интегральные характеристики модулярного кода и декодирующие процедуры восстановления секрета-оригинала в пороговых криптосхемах. На основе положений минимально избыточной модулярной арифметики (МИМА) разработан метод [7] выполнения декодирующей процедуры в пороговом устройстве разделения секрета и связанных с ней немодульных операций: расширения кода, масштабирования, деления целых чисел, преобразования модулярного кода, контроля ошибок и др., включая вычисление и использование множеств ИХМК, знаков или цифр полиадического кода. Многообразие применяемых формирователей ИХМК свидетельствует о их ключевой роли в модулярных арифметических устройствах автоматического управления. Благодаря оригинальности своей структуры они могут быть сформированы в рамках единого алгоритма [6]. Что касается машинной арифметики, то для построения любых ее вариантов как с фиксированной, так и с плавающей запятой достаточно сформировать следующий базовый набор ИХМК:

$$<x_1^-, x_2^-, \ldots, x_k^-; \hat{I}_k(X); \theta(X), \theta^-(X)>, \quad (1)$$

где x_i^- – i-я цифра симметрического полиадического кода ($i = 1, 2, \ldots, k$); $\hat{I}_k(X)$ – машинный интервальный индекс числа x; $\theta(X)$ и $\theta^-(X)$ – поправки Амербаева, соответствующие числу X во вспомогательной МСС с основаниями $m_1, m_2, \ldots, m_{k-1}, m_0$; X – произвольный элемент рабочего диапазона с попарно простыми основаниями $m_1, m_2, \ldots, m_k$ ($m_k > 2m_0 + \rho$, $m_0 \geq \rho$).

Базовый набор ИХМК (1) может либо расширяться за счет включения в него характеристик, являющихся производными базовых, либо применяться в сокращенном по сравнению с базовым виде, оставляя в нем, например, только одну интегральную характеристику $\hat{I}_k(X)$ – интервальный индекс.

Пороговая схема разделения секрета. В работе [8] разработана реализация (k, n)-пороговой схемы Ади Шамира [9]. Ее функциональной особенностью является интерполяция многочлена с коэффициентами из заданного поля Галуа с p элементами, которые становятся долевыми секретами участников эксперимента. В предложенной в [8] реализации работа схемы осуществляется в два этапа. На первом этапе дилер генерирует $(k-1)$ элементов из заданного поля, которые становятся коэффициентами многочлена $F(i)$, где $i \in (1, \ldots, k = l)$. На втором этапе каждому из n участников назначается не равный нулю номер и дилер формирует его долю секрета – пару $(i, F(i))$, где i – порядковый номер участника, а $F(i)$ – значение многочлена в соответствующей точке.

Достоинством схемы Шамира считается масштабируемость, так как количество участников можно увеличивать до порядка поля p. При этом не меняется способность восстанавливать секрет. С учетом специфики задач восстановления секрета важна временна́я составляющая этой способности.

Рассмотрим множество $Z_m \equiv (0, 1, \ldots, m-1)$, каждый элемент $\chi = \left|\frac{A}{B}\right|_m$ которого удовлетворяет сравнению $B_\chi \equiv A \pmod{m}$ (A и B – целые числа, $B_m \neq 0$). Система сравнения, формирующая множество всех целых чисел X и соответствующих кодов, определяется следующим образом:

$$\begin{cases} X \equiv \chi_1 \bmod m_1, \\ X \equiv \chi_2 \bmod m_2, \\ \ldots\ldots\ldots\ldots \\ X \equiv \chi_k \bmod m_k. \end{cases} \quad (2)$$

В МСС с основаниями ($m_1, m_2, \ldots, m_s$) ($s > 1$) модулярный код целых чисел X представляется в виде

$$(\chi_1, \chi_2, \ldots, \chi_s) = (|X|_{m_1}, |X|_{m_2}, \ldots, |X|_{m_S}). \quad (3)$$

В пороговой МА-криптосистеме разделения секрета наиболее трудоемкой является операция восстановления секрета-оригинала. Фундаментальные особенности МИМА, включая ее интегрально-аппаратную составляющую, позволяют минимизировать временны́е затраты на выполнение трудоемких немодульных операций и относительно сложную операцию восстановления секрета-оригинала по модулярным кодам маскирующих аналогов групп абонентов.

Результаты и обсуждение. Пороговые МИМА-схемы разделения секрета существуют в двух вариантах. К первому относятся системы, работающие с абонентами, число l которых не меньше порогового значения t, а ко второму – системы, в которых группа абонентов имеет численность $k < t$. Приведенные далее рассуждения применимы для обоих вариантов.

Особенность использования минимально избыточного кодирования для пороговой (t, n)-криптосхемы разделения секрета состоит в следующем [6, 10].

Основания МСС характеризуются перечнем значений m_i (4), определяемых выбранным базисом, а цифры $\tilde{\sigma}_j$, определяемые уравнением (5), рассматриваются как долевые секреты:

$$m_i = P_{I_1} = \prod_{i=1}^{i} p_i \ (i = \overline{1, n}); \qquad \tilde{\sigma}_j = |\tilde{s}|_{mj} \ (j = \overline{1,l}). \tag{4}$$

Над секрет-оригиналом S в МСС с базисом P выполняется маскирующее преобразование с помощью простой в реализации линейной функции следующего вида:

$$\tilde{S} = S + C \cdot p, \tag{5}$$

где C – псевдослучайная целочисленная величина из множества, порождающего кратные модулю P целые числа. Имеет место сужение области изменения маскирующего аналога $\tilde{S}$. При этом, естественно, сокращаются объем и требуемое время декодирования секрета. Реализация при выбранном базисе $p_{\_1}$, $p_{\_2}$, …, $p_{\_n}$ минимально избыточного кодирования сводит к теоретическому минимуму сложность расчета базовой интегральной характеристики.

Применение величины C в маскирующей функции вносит вариационную аддитивную компоненту псевдослучайного типа так, что она кратна модулю P выбранного базиса оснований МСС.

Интервальный индекс $I_l(\tilde{s})$ числа $\tilde{s} = (\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_l)$ в МСС с базисом $\{m_1, m_2, \ldots, m_l\}$ определяется позиционной формой числа $\tilde{s}$ по базису M, вычисляемому согласно принципу минимального избыточного модулярного кодирования (теорема 2 [10]):

$$\tilde{s} = \sum_{j=1}^{l-1} M_{j,l-1} \left| M^{-1}_{i,l-1} \tilde{\sigma}_J \right|_{m_j} + M_{l-1} I_l(\tilde{s}). \tag{6}$$

Разделяемый <n> сторонами исходный секрет представляет собой целое число $S \in Z_P$, где P – большой модуль, взаимно простой с $p_1, p_2 \ldots, p_n$:

$$\begin{gathered} P_{i\_l} = \prod_{i=1}^{l} P_{il}; \\ P_{i\_k} = \prod_{i=1}^{k} P_{ik}; \\ P_{I\_l} = \{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\}; \\ P_{I\_k} = \{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\}. \end{gathered} \tag{7}$$

Долевыми частичными секретами одноименных абонентов считаются следующие величины:

$$\tilde{\Sigma}_i = |\tilde{s}|_{p_i} = \left| \sigma_i + |C_p|_{p_i} \right|_{p_i} \ (\sigma_i = S_{p_i}, \ i = \overline{1, n}). \tag{8}$$

Восстанавливать секрет-оригинал S можно только по маскирующим частичным секретам абонентов (t, n)-пороговой системы разделения секрета маскирующей функции $\tilde{s}$. Предусматривается согласование диапазона изменения псевдослучайного параметра $<C>$ и области изменения оригинала сигнала. Это позволяет применять минимально избыточную модулярную декомпозицию функции маскирования при любом допустимом базисе оснований схемы.

Интервальный индекс $I_l(\tilde{s})$ числа $\tilde{s} = (\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_l)$ диапазона $\{\_P_{t-1}, \_P_{t-1} + 1, \ldots, p_0 P_{t-1} - 1\}$, где p_0 – вспомогательный модуль в МСС с базисом $\{m_1, m_2, \ldots, m_l\}$, полностью определяется компьютерным вычетом интервального индекса $\hat{I}_l(\tilde{s}) = | I_l(\tilde{s}) |_{m_l}$ только в случаях безусловного выполнения следующего требования [7]:

$$m_l \geq p_0 + l - 2 \ (p_0 \geq 2);$$

$$I_l(\tilde{s}) = \begin{cases} \hat{I}_l(\tilde{s}), \text{ если } \hat{I}_l(\tilde{s}) < p_0; \\ \hat{I}_l(\tilde{s}) - m_l, \text{ если } \hat{I}_l(\tilde{s}) \geq p_0; \end{cases}$$

$$\hat{I}_l(\tilde{s}) = \left| \sum_{j=1}^{l} R_{j,l}(\tilde{\sigma}_J) \right|_{m_l}; \ R_{j,l}(\tilde{\sigma}_j) = \left| M_{l-1}^{-1} \tilde{\sigma}_j \right|_{m_l}; \tag{9}$$

$$R_{j,l}(\tilde{\sigma}_J) = \left| -m_j^{-1} \left| M_{J,l-1}^{-1} \tilde{\sigma}_j \right|_{m_j} \right|_{m_l} \ (j \neq l).$$

Использование интервально-модулярной формы числа $\overline{S}$ снижает сложность расчета базовых интервально-индексных характеристик при решении задач порогового кодирования практически до теоретического минимума.

Процедуру выбора рабочего диапазона изменения секрета-маски необходимо совмещать с поиском условий, обеспечивающих непересекаемость диапазонов (множеств) целых чисел $\tilde{S}_{I\_i} \pmod{P_{I\_i}}$ [10].

Корректность пороговых МА-криптосистем разделения секрета во многом зависит от оптимальности множества псевдослучайных целочисленных величин, используемых при проведении маскирующего преобразования (5). Оптимизация может выполняться по мощности этого множества, его структуре или другим характеристикам. Важнейшим оптимизационным аспектом рассмотренной проблемы синтеза модулярной пороговой (t, n)-криптосхемы разделения секрета является минимизация мощности $|C_p|$ множества $C_p \in (\tilde{S}_{\text{НП}}, \tilde{S}_{\text{ВП}})$, где $\tilde{S}_{\text{НП}}$ и $\tilde{S}_{\text{ВП}}$ – нижнее и верхнее значения секрета. Параметры множества C_p рассчитываются предварительно и записываются в память.

Заключение. Результаты представленной работы позволяют сделать следующие выводы для модулярных пороговых криптосистем разделения секрета в распределенных системах обработки данных:

1. Применение линейной маскирующей функции и сужение области изменения маскирующего аналога $\tilde{S}$ секрета-оригинала S, допускающее при выбранных $p\_1, p\_2, \ldots, p\_n$ минимально избыточное кодирование, обусловливает существенное снижение вычислительной сложности расчетных соотношений МИМА интегральных характеристик в рамках исследуемой модели, благодаря чему достигается более высокий уровень производительности на стадии декодирования секрета-оригинала по сравнению с другими решениями.

2. Адаптивное согласование диапазона изменений псевдослучайного параметра маскирующей функции с областью ее значений позволяет реализовать минимально избыточную модулярную декомпозицию функции маскирования при любом допустимом базисе оснований схемы.

3. Для оптимального решения проблемы синтеза модулярной пороговой (t, n)-криптосхемы разделения секрета необходимо минимизировать мощность множества $C_p \in (\tilde{S}_{\text{НП}}, \tilde{S}_{\text{ВП}})$, где $\tilde{S}_{\text{НП}}$ и $\tilde{S}_{\text{ВП}}$ – нижнее и верхнее значения секрета.

Вклад авторов. *А. Ф. Чернявский* и *А. А. Коляда* разработали концепцию и основные положения работы, *А. Ф. Чернявский, Е. И. Козлова* и *В. С. Садов* провели критический анализ содержания статьи и подготовили окончательный вариант работы для публикации.

Список использованных источников

1. Артюхов, Ю. В. Анализ схем разделения секрета, использующих вероятностный и комбинаторный подход в реализации пороговых криптосистем, функционирующих в распределенных компьютерных системах / Ю. В. Артюхов // Актуальные вопросы технических наук : материалы Междунар. заоч. науч. конф., Пермь, июль 2011 г. / под общ. ред. Г. Д. Ахметовой. – Пермь : Меркурий, 2011. – 80 с.

2. Коблиц, Н. Курс теории чисел и криптографии / Н. Коблиц ; пер. с англ. М. А. Михайловой, В. Е. Тараканова. – М. : Научное издательство ТВП, 2001. – 254 с.

3. Носиров, З. А. Анализ криптографических схем разделения секрета для резервного хранения ключевой информации / З. А. Носиров, О. В. Щербинина // Прикаспийский журнал: управление и высокие технологии. – 2019. – № 2(46). – URL: https://cyberleninka.ru/article/n/analiz-kriptograficheskih-shem-razdeleniya-sekreta-dlya-rezervnogo-hraneniya-klyuchevoy-informatsii (дата обращения: 06.01.2025).

4. Модулярная арифметика и ее приложения в инфокоммуникационных технологиях / Н. И. Червяков, А. А. Коляда, П. А. Ляхов [и др.]. – М. : Физматлит, 2017. – 400 с.

5. Харин, Ю. С. Математические и компьютерные основы криптологии : учеб. пособие / Ю. С. Харин. – Минск : Новое знание, 2003. – 382 с.

6. Коляда, А. А. Модулярные структуры конвейерной обработки цифровой информации / А. А. Коляда, И. Т. Пак. – Минск : Университетское, 1992. – 256 с.

7. Чернявский, А. Ф. Особенности машинной арифметики высокопроизводительных модулярных вычислительных структур / А. Ф. Чернявский, Е. И. Козлова, А. А. Коляда // Журнал Белорусского государственного университета. Математика. Информатика. – 2023. – № 2. – С. 94–101.

8. Виноградова, А. А. Системы разделения секрета / А. А. Виноградова. – 2017. – 19 с. – URL: http://hdl.handle.net/11701/11134 (дата обращения: 27.01.2025).

9. Shamir, A. How to share a secret / A. Shamir // Communications of the ACM. – Nov. 1979. – Vol. 22, iss. 11. – P. 612–613. – DOI: 10.1145/359168.359176.

10. Коляда, А. А. Пороговый метод разделения секрета на базе избыточных модулярных вычислительных структур / А. А. Коляда, П. В. Кучинский, Н. И. Червяков // Информационные технологии. – 2019. – Т. 25, № 9. – С. 553–560.

References

1. Artjukhov Yu. V. *Analysis of secret sharing schemes using probabilistic and combinatorial approaches in the implementation of threshold cryptosystems operating in distributed computer systems*. Aktual'nye voprosy tehnicheskih nauk : materialy Mezhdunarodnoj zaochnoj nauchnoj konferencii, Perm', ijul' 2011 g. [*Current Issues in Technical Sciences : Materials of the International Correspondence Scientific Conference, Perm, July 2011*]. In G. D. Akhmetova (ed.). Perm’, Mercurii, 2011, 80 p. (In Russ.).

2. Koblitz N. *A Course in Number Theory and Cryptography*, second edition. Springer, 1994, 245 p.

3. Nosirov Z. A., Shcherbina O. V. *Analysis of cryptographic secret sharing schemes for backup storage of key information*. Prikaspiiskii zhurnal: upravlenie i vysokie tekhnologii [*Caspian Journal: Management and High Technologies*], 2019, no. 2(46) (In Russ.). Available at: https://cyberleninka.ru/article/n/analiz-kriptograficheskih-shem-razdeleniya-sekreta-dlya-rezervnogo-hraneniya-klyuchevoy-informatsii (accessed 06.01.2025).

4. Chervyakov N. I., Kolyada A. A., Lyahov P. A., Babenko M. G., Lavrinenko I. N., Lavrinenko A. V. Moduljarnaja arifmetika i ee prilozhenija v infokommunikacionnyh tehnologijah. *Modular Arithmetic and its Applications in Infocommunication Technologies*. Moscow, Fizmatlit, 2017, 400 p. (In Russ.).

5. Kharin Yu. S. Matematicheskie i komp'juternye osnovy kriptologii. *Mathematical and Computer Foundations of Cryptology*. Minsk, Novoe znanie, 2003, 382 p. (In Russ.).

6. Kolyada A. A., Pak I. T. Modulyarnye struktury konveyernoy obrabotki tsifrovoy informatsii. *Modular Structures of Pipeline Processing of Digital Information*. Minsk, Universitetskoe, 1992, 256 p. (In Russ.).

7. Chernyavskiy A. F., Kozlova E. I., Kolyada A. A. *Features of machine arithmetic of high-performance modular computing structures.* Zhurnal Belorusskogo gosudarstvennogo universiteta. Matematika. Informatika [*Journal of the Belarusian State University. Mathematics and Informatics*], 2023, no. 2, pp. 94–101 (In Russ.).

8. Vinogradova A. A. Sistemy razdeleniya sekreta. *Secretion Separation Systems*, 2017, 19 p. (In Russ.). Available at: http://hdl.handle.net/11701/11134 (accessed 27.01.2025).

9. Shamir A. How to share a secret. *Communications of the ACM*, November 1979, vol. 22, iss. 11, pp. 612–613. DOI: 10.1145/359168.359176.

10. Kolyada A. A., Kuchinskiu P. V., Chervyakov N. I. *Threshold secret sharing method based on redundant modular computing structures*. Informatsionnye tekhnologii [*Information Technology*], 2019, vol. 25, no. 9, pp. 553–560 (In Russ.).

Информация об авторах

Чернявский Александр Федорович, доктор технических наук, академик НАН Беларуси, профессор, Белорусский государственный университет.
E-mail: ChernAA@bsu.by

Козлова Елена Ивановна, кандидат физико-математических наук, доцент, Белорусский государственный университет.
E-mail: kozlova@bsu.by

Садов Василий Сергеевич, кандидат технических наук, доцент, Белорусский государственный университет.
E-mail: sadov@bsu.by

Коляда Андрей Алексеевич, доктор физико-математических наук, Белорусский государственный университет.

Information about the authors

Alexander F. Chernyavskiy, D. Sc. (Eng.), Acad. of the National Academy of Sciences of Belarus, Prof., Belarusian State University.
E-mail: ChernAA@bsu.by

Alena I. Kazlova, Ph. D. (Phys.-Math.), Assoc. Prof., Belarusian State University.
E-mail: kozlova@bsu.by

Vasiliy S. Sadov, Ph. D. (Eng.), Assoc. Prof., Belarusian State University.
E-mail: sadov@bsu.by

Andrei A. Kolyada, D. Sc. (Phys.-Math.), Belarusian State University.

ИНФОРМАЦИЯ
INFORMATION

ОРГАНИЗАТОРЫ

- Национальная академия наук Беларуси.
- Агентство по космическим исследованиям Национальной академии наук Беларуси.
- Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси».

ЦЕЛИ КОНГРЕССА

- Обсуждение новых достижений в космической области, определение приоритетных направлений развития космической деятельности.
- Укрепление международного сотрудничества в области космических исследований и использования космического пространства в мирных целях.
- Расширение совместных усилий белорусских и зарубежных деловых кругов, фирм, компаний, ученых и специалистов, направленных на выполнение космических программ и проектов, анализ хода их реализации и выбор перспективных направлений дальнейших исследований.
- Развитие партнерства правительственных и общественных организаций, предприятий и организаций разных форм собственности по использованию космических средств и технологий в интересах различных отраслей экономики.

ТЕМАТИКА КОНГРЕССА

- Инновационные программы, проекты и технологии в ракетно-космической отрасли. Использование результатов космической деятельности в интересах различных отраслей экономики.
- Космические аппараты, целевая и научная аппаратура. Системы навигационно-временного обеспечения, спутниковой связи и вещания.
- Средства, технологии и методы обработки и отображения данных дистанционного зондирования Земли, геосервисы на их основе. Искусственный интеллект в космических технологиях.
- Технологии обучения и подготовки кадров для космической отрасли.
- Теплофизические аспекты практической космонавтики, перспективные материалы, элементы и устройства для космической техники.
- Космические исследования в области биологии, физиологии и медицины.

РАБОЧИЕ ЯЗЫКИ КОНГРЕССА

Белорусский, русский, английский

ВАЖНЫЕ ДАТЫ

Подача заявки на участие и представление доклада	–	**до 13 июня 2025 г.**
Извещение о приеме доклада	–	**до 1 августа 2025 г.**
Оплата организационного взноса по безналичному расчету	–	**до 25 сентября 2025 г.**

ОБЩИЕ ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ И ПРЕДСТАВЛЕНИЮ ДОКЛАДОВ

Доклады на конгресс принимаются в электронном виде на белорусском, русском или английском языках по электронной почте: belcongress9@yandex.by. Белорусские авторы в обязательном порядке предоставляют также в оргкомитет конгресса акт экспертизы о возможности опубликования материалов в открытой печати (нарочным или письмом на почтовый адрес конгресса). Без данного документа доклад оргкомитетом не рассматривается. Труды конгресса будут изданы к началу его работы.

Объем представляемого доклада не должен превышать четырех полных ненумерованных страниц формата А4. При форматировании текста устанавливаются поля по 2,5 см со всех сторон, шрифт текста рукописи Times New Roman, размер шрифта основного текста 12 пт, интервал между строками одинарный, абзацный отступ 0,7 см. Доклад должен включать:

УДК (в левом верхнем углу, размер шрифта 11 пт);

НАЗВАНИЕ (без переносов, прописными буквами полужирным начертанием 14 пт);

И. О. Фамилия авторов (инициалы имени и отчества с отбивкой, затем фамилия);

для каждого автора полное официальное название организации, город, страну (если в подготовке доклада принимали участие авторы из разных учреждений, необходимо указать принадлежность каждого автора[1] к конкретному учреждению[1] с помощью надстрочного индекса;

аннотацию (не более 100 слов, без аббревиатур и формул, 11 пт, курсив);

основной текст доклада (кроме английского языка) с переносами (при наличии рубрикаций они нумеруются, выделяются полужирным шрифтом и отбиваются от основного текста), включая иллюстрации, формулы и таблицы.

Требования и примеры по оформлению докладов на русском языке представлены на сайте конгресса http://sit.basnet.by/congress9.

Авторам докладов необходимо зарегистрироваться на сайте конгресса и указать полностью фамилию, имя и отчество, организацию, должность, ученую степень и ученое звание, почтовый адрес, телефоны, факс и e-mail для оперативной связи.

ИНФОРМАЦИЯ О КОНГРЕССЕ

Девятый Белорусский космический конгресс будет проводиться в Минске в Объединенном институте проблем информатики НАН Беларуси в период **с 21 по 23 октября 2025 г.**

Для участия в работе конгресса необходимо **до 25 сентября 2025 г**. оплатить организационный взнос по безналичному расчету. Информация об уплате организационного взноса и реквизиты для оплаты будут предоставлены авторам принятых докладов.

Лица, участвующие в работе конгресса в качестве слушателя, должны зарегистрироваться на сайте конгресса. Организационный взнос слушатели не уплачивают.

ПОЧТОВЫЙ АДРЕС КОНГРЕССА

Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси» (ОИПИ НАН Беларуси), ул. Сурганова, 6, г. Минск, 220012, Беларусь.

КОНТАКТЫ:

Тел.: +375 17 275 20 74 Кореняко Сергей Анатольевич
+375 17 374 20 75 Морозов Дмитрий Васильевич
+375 17 224 26 02 Ляткевич Ирина Антоновна
Факс: +375 17 270 31 75
E-mail: belcongress9@yandex.by
Официальный сайт: http://sit.basnet.by/congress9

Правила для авторов

Редакция журнала «Информатика» просит авторов руководствоваться приведенными ниже правилами.

I. Статьи принимаются в редакцию через электронную систему подачи по адресу http://inf.grid.by в формате файлов текстовых редакторов Microsoft Word. Объем оригинальной статьи – от 8 до 16 стр., включая рисунки, таблицы и достаточное количество наиболее актуальных ссылок; объем обзорной статьи – от 16 до 32 стр., включая все основные ссылки. Текст набирается с переносами, шрифт Times New Roman 11 пт, интервал между строками одинарный, абзацный отступ 0,5 см, поля по 2,5 см со всех сторон.

Материал статьи должен быть четко структурированным: Введение; основные разделы, в которых изложены цели и задачи, методы, результаты; Заключение (выводы).

II. Статьи о результатах работ, проведенных в научных учреждениях, должны иметь разрешение на публикацию (сопроводительное письмо за подписью руководителя или выписку из заседания ученого совета, отдела или кафедры, акт экспертизы).

III. Статьи в обязательном порядке должны включать аннотацию, ключевые слова, список литературы, информацию об авторах на русском и английском языках.

На титульной странице располагаются следующие метаданные:

1. Индекс по универсальной десятичной классификации (УДК); *на русском и английском языках* тип статьи (оригинальная или обзорная), название статьи, инициалы и фамилии всех авторов, полное наименование учреждений, где работают авторы, с указанием почтового адреса, при наличии указывается ученая степень и ORCID, e-mail ответственного лица.

2. Аннотация (Abstract) объемом 150–250 слов в оригинальной статье должна быть структурирована отдельными подразделами: Цели, Методы, Результаты, Заключение, а также максимально характеризовать содержательную часть рукописи. Сюда не следует включать впервые введенные термины, аббревиатуры (за исключением общеизвестных), ссылки на литературу.

3. Ключевые слова (Keywords) – наиболее значимые слова или словосочетания по теме работы, отражающие специфику темы, объекты и результаты исследования; перечень ключевых слов должен содержать 5–10 слов.

4. В разделе Благодарности (Acknowledgements) указываются все источники финансирования исследования, а также благодарности людям, которые участвовали в работе над статьей.

5. Автор обязан уведомить редакцию о реальном или потенциальном конфликте интересов, включив информацию в раздел Конфликт интересов (Conflict of interest).

6. Формулы, рисунки, таблицы в статье нумеруются в соответствии с порядком их упоминания в тексте. Ссылки на рисунки и таблицы в тексте обязательны. Рисунки должны быть выполнены с хорошим разрешением в масштабе, позволяющем четко различать надписи и обозначения. Цветные иллюстрации печатаются только в том случае, когда это необходимо для понимания излагаемого материала. Подрисуночные подписи с расшифровкой всех позиций, представленных на рисунке, и названия таблиц набираются шрифтом гарнитуры основного текста размером 9 пт. Перевод подрисуночной подписи и пояснений к рисунку, а также перевод названия таблицы, заголовки строк или столбцов располагаются курсивом после русскоязычной версии.

7. Набор формул выполняется в формульном редакторе Microsoft Equation или Math Type. Прямым шрифтом набираются: греческие и русские буквы; математические символы (sin, lg, ∞); символы химических элементов (C, Cl, CHCl3); цифры (римские и арабские); индексы (верхние и нижние), являющиеся сокращениями слов. Курсивом набираются латинские буквы, символы физических величин (в том числе и в индексе).

8. Список использованной литературы оформляется в соответствии с требованиями Высшей аттестационной комиссии Республики Беларусь (ГОСТ 7.5–2008). Номер литературной ссылки в тексте дается порядковым номером в квадратных скобках. Ссылаться на неопубликованные работы не допускается.

9. Отдельно оформляется References со следующей структурой: авторы (транслитерация), транслитерированное название монографии, *Перевод названия монографии на английский язык*. Выходные данные с обозначениями на английском языке. От транслитераций названий статей можно отказаться.

10. Ссылки на учебно-методическую литературу, ГОСТы, авторефераты, статистические отчеты в список не включаются, а оформляются в виде сносок (с подробными рекомендациями можно ознакомиться на сайте журнала в разделе Правила для авторов).

11. В разделе Информация об авторах (Information about the authors) приводятся ФИО авторов полностью, ученая степень, звание, должность, название организации, ORCID (при наличии).

IV. Все поступающие в редакцию рукописи проходят предварительную проверку на соответствие Правилам для авторов. Статья может быть возвращена автору на доработку с просьбой устранить недостатки или дополнить информацию. После проверки на соответствие правилам статья направляется рецензенту с указанием сроков рецензирования.

V. При наличии замечаний рецензента автору предоставляется определенное время на доработку рукописи. Статьи, направляемые на доработку, должны быть возвращены в исправленном виде с ответами на все замечания. Окончательное решение о публикации или отклонении рукописи принимается редколлегией журнала. При положительном заключении рецензента статья передается редактору для подготовки к печати. Редакция оставляет за собой право на редакционные изменения, не искажающие основное содержание статьи.

VI. Редакция журнала предоставляет возможность первоочередного опубликования статей, представленных лицами, которые осуществляют послевузовское обучение (аспирантура, докторантура, соискательство) в год завершения обучения.

VII. Авторы несут ответственность за направление в редакцию статей, уже опубликованных ранее или принятых к публикации другими изданиями.

ИНДЕКСЫ

00827
для индивидуальных
подписчиков

008272
для предприятий
и организаций

ISSN 1816-0301 (Print)

ИНФОРМАТИКА ■ INFORMATICS ТОМ ■ VOL. 22 1 | 2025