ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
СПбГУТ))) Июнь 2017 Том 5 № 2 ISSN 2307-1303 http://www.itt.sut.ru
TELECOM IT
June 2017 Vol. 5 Iss. 2 ISSN 2307-1303 http://www.itt.sut.ru

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ АВТОМАТИЧЕСКОГО ПОСТРОЕНИЯ УМНОЖИТЕЛЕЙ ЭЛЕМЕНТОВ ДВОИЧНОГО ПОЛЯ ГАЛУА

С. С. Владимиров[1*], Д. Ф. Мухаметшина[1]

[1] СПбГУТ, Санкт-Петербург, 193232, Российская Федерация
* Адрес для переписки: vladimirov.opds@gmail.com

Аннотация
Разработка современных цифровых систем передачи данных требует применять алгоритмы шифрования и помехоустойчивого кодирования, которые основаны на вычислениях в конечных полях Галуа большой степени. При этом основными базовыми операциями являются умножение и сложение. Операция умножения требует большого количества логических элементов при аппаратной реализации и большого числа тактов процессора при программной реализации, что приводит к значительным задержкам вычислений при реализации ресурсоёмких алгоритмов декодирования и шифрования. Однако многие алгоритмы умножения элементов конечного поля имеют регулярную структуру, позволяющую автоматизировать процедуру формирования аппаратных схем и генерации программного кода умножителей. **Предмет исследования.** Статья посвящена разработанной авторами системе автоматического построения умножителей элементов двоичного поля Галуа. Отдельно рассматривается вопрос сравнения быстродействия умножителей разных типов с использованием данной системы. **Метод.** Рассмотрена структура разработанного программного обеспечения. Проведено моделирование программных процессов, позволившее сравнить быстродействие основных умножителей элементов поля. **Основные результаты.** Проведено сравнение ряда популярных алгоритмов умножения элементов поля Галуа с точки зрения быстродействия их программных реализаций, и даны рекомендации по выбору алгоритмов построения умножителей при разработке систем передачи данных. **Практическая значимость.** Разработанное программное обеспечение позволяет автоматизировать процедуру разработки систем помехоустойчивого кодирования и шифрования.

Ключевые слова
автоматизация разработки, автоматическое генерирование кода, поля Галуа, умножение элементов поля Галуа, умножитель Рейхани-Мазолеха, умножитель Карацубы-Офмана.

Информация о статье
УДК 004.4'242
Язык статьи – русский.
Поступила в редакцию 05.04.17, принята к печати 02.06.17.

Ссылка для цитирования: Владимиров С. С., Мухаметшина Д. Ф. Программный комплекс для автоматического построения умножителей элементов двоичного поля Галуа // Информационные технологии и телекоммуникации. 2017. Том 5. № 2. С. 62–73.

THE SOFTWARE FOR AUTOMATIC DEVELOPING OF MULTIPLIERS OVER BINARY GALOIS FIELD

S. Vladimirov[1*], D. Mukhametshina[1]

[1] SPbSUT, St. Petersburg, 193232, Russian Federation
* Corresponding author: vladimirov.opds@gmail.com

Abstract—The development of modern digital data transmission systems requires the use of encryption and error-correcting coding algorithms, which are based on calculations in finite Galois fields of a large degree. In this case, the basic operations are multiplication and addition. The operation of multiplication requires a large number of logical elements in the hardware implementation and a large number of processor cycles in software implementation, which leads to significant delays in the computation of resource-intensive decoding and encryption algorithms. However, many algorithms for multiplying the elements of a finite field have a regular structure, which makes it possible to automate the procedure of both the formation of the hardware circuit and the generation of multipliers program code. **Research subject.** The article is devoted to the authors's developed automated source code generation software for creating binary Galois field elements multipliers. The question of comparing the speed of multipliers of different types using this system is considered separately. **Method.** The structure of the developed software is considered. Modeling of software processes was carried out, which made it possible to compare the speed of the main multipliers of field elements. **Core results.** A number of popular algorithms for multiplying the elements of the Galois field from the viewpoint of the speed of their software implementations are compared, and recommendations are given on the choice of algorithms for constructing multipliers in the development of data transmission systems. **Practical relevance.** The developed software allows to automate the procedure for developing systems of error-correcting coding and encryption.

Keywords—development automation, automatic source code developing, Galois field, Galois field elements multiplication, Reyhani-Masoleh multiplier, Karatsuba multiplier.

Article info
Article in Russian.
Received 05.04.17, accepted 02.06.17.

For citation: Vladimirov S., Mukhametshina D.: The Software for Automatic Developing of Multipliers over Binary Galois Field // Telecom IT. 2017. Vol. 5. Iss. 2. pp. 62–73 (in Russian).

Введение

Современные сети и системы цифровой передачи данных невозможно представить без использования криптографических алгоритмов, обеспечивающих конфиденциальность и целостность передаваемой информации, и алгоритмов помехоустойчивого кодирования, повышающих достоверность при передаче информационных сообщений по каналам связи с помехами. Многие популярные методы помехоустойчивого кодирования и шифрования основаны на вычислениях в конечных двоичных полях Галуа. К примеру алгоритм симметричного шифрования AES (стандарт FIPS 197)[1] [1], блочный шифр «Кузнечик» (ГОСТ Р 34.12–

[1] FIPS 197. Advanced Encryption Standard (AES). 2001-09-26. National Institute of Standards and Technology. 2001. 51 p.

2015)[2] и функция хэширования «Стрибог» (ГОСТ Р 34.11–2012)[3] основаны на вычислениях в поле GF(2^8). Среди помехоустойчивых кодов, построенных на базе математики полей Галуа, нельзя не отметить коды Боуза–Чоудхури–Хоквингема (БЧХ) и коды Рида–Соломона (РС) [2], которые используются раздельно, например, в международной спутниковой поисково-спасательной системе Cospas-Sarsat применены укороченные коды БЧХ (82, 61) над полем GF(2^7) и (38, 26) над полем GF(2^6)[4], или в составе каскадных кодов, например, в технологии беспроводной связи WiMAX 802.16d код РС (255, 239) над полем GF(2^8) используется в качестве внешнего кода[5]. В большинстве алгоритмов шифрования/дешифрования и кодеках помехоустойчивых кодов используются две основные операции над полем Галуа — сложение и умножение. Операция умножения представляет наибольшую сложность, приводя к значительным задержкам при реализации алгоритмов декодирования, например, синдромного декодирования по алгоритму Берлекэмпа–Мэсси [2] или мажоритарного декодирования на основе двойственного базиса [3, 4, 5]. На практике применяется значительное число алгоритмов умножения элементов полей Галуа, среди которых можно выделить метод умножения, основанный на операциях логарифмирования и антилогарифмирования и требующий хранения в памяти двух массивов элементов поля, классический алгоритм умножения «сдвиг-со-сложением, справа-налево», алгоритм Карацубы–Офмана [6, 7, 8], предназначенный для умножения длинных чисел, а также различные варианты алгоритма Мастровито, например, так называемый алгоритм Рейхани-Мазолеха, предложенный в начале 2000-х гг. членами IEEE А. Рейхани-Мазолехом и М. А. Хасаном [8, 9]. При использовании конечных полей Галуа больших степеней, задача построения умножителя по любому из приведенных алгоритмов отличается значительной сложностью построения аппаратной схемы умножителя и написания программного кода. Тем не менее, такие алгоритмы имеют регулярную структуру, которая позволяет автоматизировать процедуру генерации программного кода умножителей и, для некоторых алгоритмов, процедуру формирования аппаратной схемы.

В рамках решаемой на кафедре сетей связи и передачи данных Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича задачи автоматизации процесса разработки систем кодирования и шифрования, основанных на операциях в конечных полях, авторами был разработан программный комплекс для автоматического формирования программного кода умножителей на нескольких языках программирования и языках описания аппаратуры.

К разработанному программному обеспечению применялись следующие требования:

1. Генерация кода на наиболее распространенных языках программирования общего применения и языках описания аппаратуры.

[2] ГОСТ Р 34.12–2015. Информационная технология. Криптографическая защита информации. Блочные шифры. 2015-06-19. М.: Стандартинформ. 2015. IV, 21 с.
[3] ГОСТ Р 34.11–2012. Информационная технология. Криптографическая защита информации. Функция хэширования. 2012-08-07. М.: Стандартинформ. 2012. IV, 34 с.
[4] Cospas-Sarsat Guidelines on 406 MHz Beacon Coding, Registration and Type Approval. C/S G.005 Issue 2 - Revision 6. 2013. 88 p.
[5] IEEE Std 802.16–2004. IEEE Standard for Local and metropolitan area networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems. 2004-06-24. NY: IEEE. 2004. 895 p.

2. Использование программы-генератора в виде консольного пользовательского приложения для персонального компьютера и в виде сетевого приложения с отдельным веб-интерфейсом.

Программа-генератор для автоматического формирования кода

Разработанное программное обеспечение реализовано на языке программирования JavaScript с использованием программной платформы Node.js[6], что обеспечивает работу программы-генератора в различных операционных системах и позволяет использовать ее ядро как в форме приложения для персонального компьютера, так и в форме веб-приложения. На разработанное программное обеспечение получено свидетельство о регистрации программы для ЭВМ № 2017618731.

Архитектурно программа-генератор представляет собой набор модулей, сформированных по следующим правилам:

1. Модули первого уровня содержат код, генерирующий реализацию алгоритма на конкретном языке.

2. Модули второго уровня содержат логику дополнительных вычислений, необходимых для работы алгоритма, и после произведения этих вычислений передают результат в модули первого уровня.

3. Модули третьего уровня отвечают за взаимодействие с пользователем, сбор входных данных и их передачу модулям второго уровня, то есть представляют собой пользовательский интерфейс.

Благодаря модульной структуре программа может быть расширена путем добавления новых алгоритмов построения умножителей или реализации существующих алгоритмов на других языках программирования. Также модульная структура позволила реализовать динамическую инициализацию именно тех модулей, которые требуются для формирования программного кода по заданным пользователем параметрам, что приводит к экономии оперативной памяти компьютера, на котором запущена программа-генератор.

Структурная схема организации программы-генератора представлена на рис. 1.

В настоящее время программа-генератор автоматически формирует программный код для умножителей, реализованных согласно алгоритмам Рейхани-Мазолеха [10, 11], «сдвиг-со-сложением, справа-налево» и Карацубы-Офмана на языках Си, Java, Pascal, Python, Verilog.

Для уменьшения требований к программно-аппаратному обеспечению платформы программа-генератор для локального компьютера реализована с использованием консольного интерфейса. При запуске программы для автоматического формирования кода умножителя пользователь задает в виде параметров командной строки исходные данные:

- алгоритм умножителя;
- язык реализации алгоритма;
- абсолютный путь к файлу, в который будет сохранен результат;
- название этого файла;

[6] Node.js. URL: https://nodejs.org/ (Accessed date: 03.03.2017).

- двоичное представление порождающего полинома поля Галуа, для которого строится умножитель.

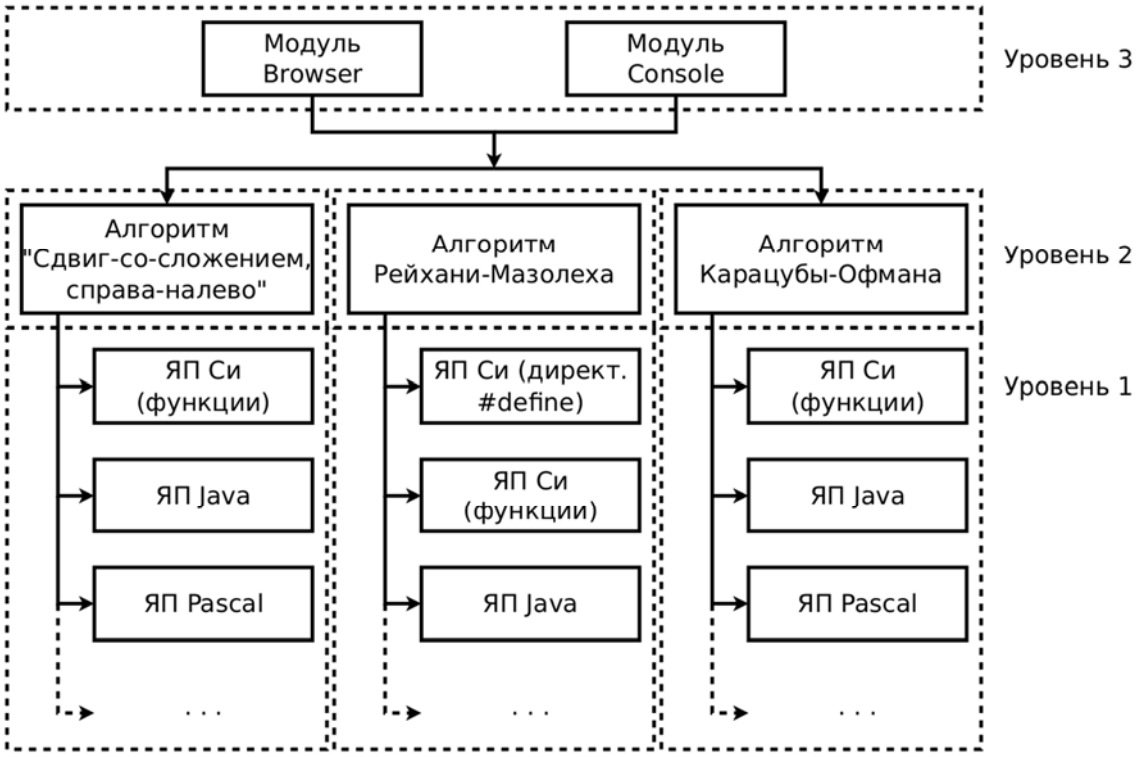

Рис. 1. Логическая структурная схема организации приложения

Обязательным параметром является только двоичное представление порождающего полинома, в котором запись бит ведется от младшего к старшему. Прочие параметры имеют значения «по умолчанию», которые приведены во встроенной справке программы, показанной на рис. 2. Показан пример выполнения программы в операционной системе Windows 7 с использованием оболочки Windows PowerShell.

```
PS D:\Dev\workspaces\js\reih-maz-multiplier> node .\src\console.js -h

  Usage: console -a karatsuba -l pascal -o D://temp/ -p 101010101

  Defaults:
        Output directory:       './tmp/'
        File name:              'out'
        Algorithm:              'reyhani-masoleh'
        Language:               'c_func'

  Supported algorithms and languages:
        karatsuba: [c_func, java, pascal, python],
        reyhani-masoleh: [c_def, c_func, java, matrix, pascal, python, verilog],
        simple: [c_func, java, pascal, python]


  Options:

    -h, --help                      output usage information
    -V, --version                   output the version number
    -a, --algorithm <algorithm>     multiplier algorithm
    -l, --language <language>       language for output code
    -n, --name <name>               name for output file
    -o, --out <absolute path>       path for output folder
    -p, --polynomial <polynomial>   monic irreducible polynomial base 2 with big-endian notation.
```

Рис. 2. Встроенная справка консольной версии программы
с указанием значений «по умолчанию»

По указанному пользователем алгоритму выбирается модуль второго уровня, а по языку — модуль первого уровня. Программа проверяет полноту и корректность ввода данных, и при необходимости указывает пользователю о необходимости внесения дополнений и предлагает варианты. Если все необходимые параметры введены корректно, то программа выводит файл программного кода с заданным названием в указанную папку. На рис. 3 приведен пример запуска программы со всеми требуемыми параметрами и результат ее выполнения. В данном случае генератор формирует программный код умножителя по алгоритму Карацубы-Офмана на языке программирования Python для поля Галуа, порожденного полиномом $p(x) = 1 + x^3 + x^4$. Показан пример выполнения программы в операционной системе GNU/Linux с использованием оболочки XTerm.

```
user@pc:~/gf-mult-gen/src/$ node ./console.js -a karatsuba -l python -o ./ -n kar -p 10011
Done
```

kar.py

```
1  class GFMultiplier:
2    def karatsuba_mult(self, A: int, B: int):
3      P = int('10011', 2)
4      result = self.modulo(self.calculate(A, B), P)
5      print('{0:b}'.format(result))
6      return result
7
8    def modulo(self, a: int, b: int):
9      B_len = len('{0:b}'.format(b))
10     A_len = len('{0:b}'.format(a))
11     while (A_len >= B_len):
12       a = a ^ (b << (A_len - B_len))
13       A_len = len('{0:b}'.format(a))
14     return a
15
16   def calculate(self, A: int, B: int):
17     n = max(len('{0:b}'.format(A)), len('{0:b}'.format(B)))
18     if n == 1:
19       return A & B
20
21     from math import ceil
22     half = ceil(n / 2)
23     full = 2 * half
24     filterHalf = (2 << half - 1) - 1
25
26     AH = A >> half
27     AL = A & filterHalf
28     BH = B >> half
29     BL = B & filterHalf
```

Рис. 3. Пример работы с программой-генератором
с использованием интерфейса командной строки

При работе в формате веб-приложения используется интерактивный графический интерфейс, реализованный на языке программирования JavaScript и языке разметки веб-страниц HTML. Для представления сгенерированного кода используется библиотека CodeMirror. На рис. 4 приведена файловая структура части веб-приложения, реализующей интерфейс пользователя.

Для работы с веб-интерфейсом можно использовать любой веб-браузер, поддерживающий язык JavaScript и отображение форм. Внешний вид веб-интерфейса показан на рис. 5 (см. ниже). Для примера рассмотрено формирование программного кода умножителя по алгоритму Рейхани-Мазолеха на языке программирования Pascal для поля Галуа, порожденного полиномом $p(x) = 1 + x^3 + x^4$.

Веб-интерфейс приложения условно разделен на три блока:
1. Форма для ввода входных данных.
2. Поле вывода сгенерированного кода.
3. Поле справки.

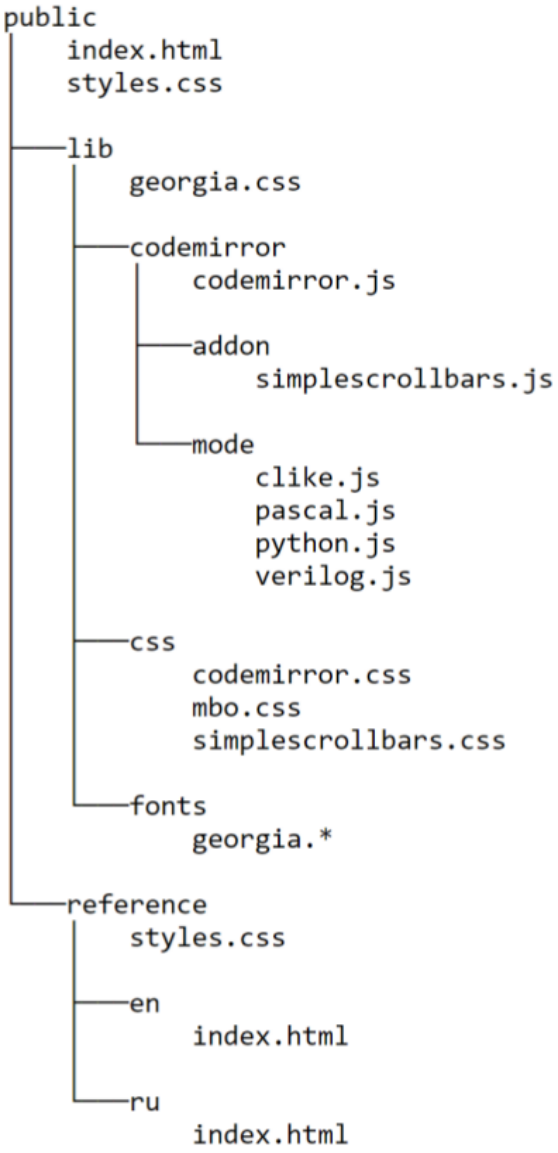

Рис. 4. Файловая структура веб-приложения (интерфейс пользователя)

В качестве входных данных пользователь задает алгоритм умножителя, язык программирования и образующий полином поля Галуа в двоичном виде. При этом для каждого алгоритма умножения выводится свой список доступных языков программирования.

Пользователь может скопировать сгенерированный код из поля вывода или скачать его на свой компьютер в виде файла с соответствующим языку программирования расширением.

Веб-приложение размещено на одном из серверов кафедры Сетей связи и передачи данных и доступно по адресу http://galois.spbsut.ru/gf-mult-gen/. Интерфейс программы представлен на двух языках: русском и английском.

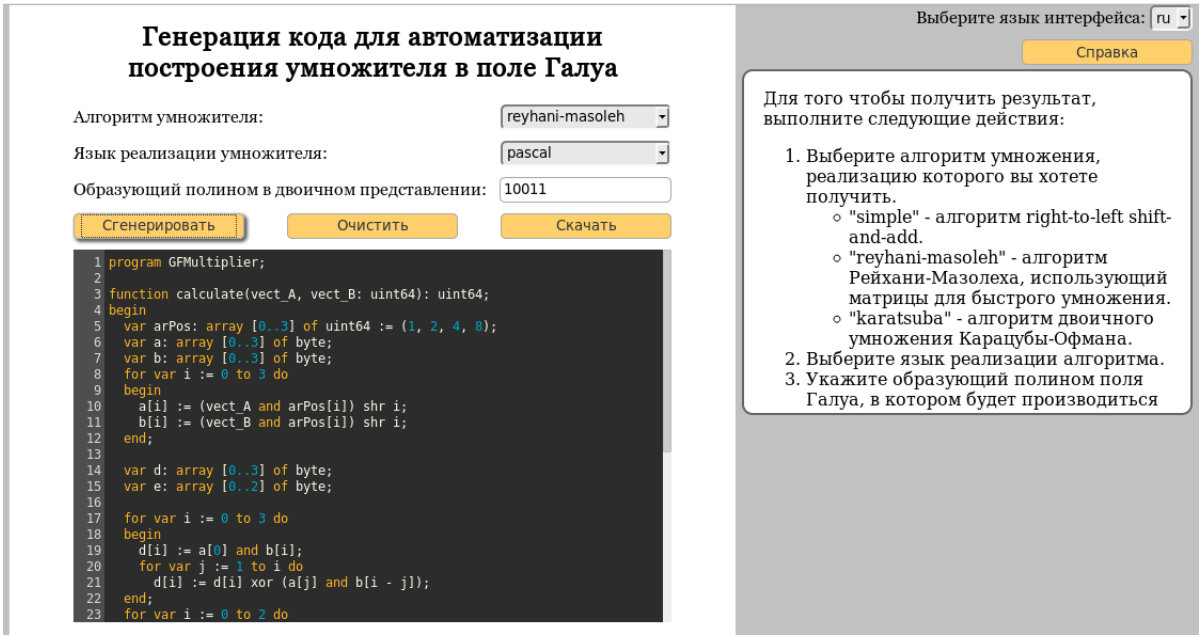

Рис. 5. Внешний вид веб-интерфейса для работы с программой-генератором

Сравнение алгоритмов умножения элементов поля

Для оценки эффективности выбранных для реализации алгоритмов умножения произведено их сравнение по времени выполнения операций умножения. Поскольку время выполнения одной операции слишком мало, и его сложно оценить при помощи стандартных средств операционной системы, измерение времени производилось для 10^8 операций умножения. Реализации алгоритмов умножения сравнивались только для языка программирования Си, поскольку на сегодня это один из наиболее популярных языков для разработки программного обеспечения систем передачи данных.

Сравнение проводилось для трех различных ЭВМ, характеристики которых приведены в таблице.

Таблица.

Характеристики ЭВМ, для которых производилось сравнение алгоритмов умножения элементов поля Галуа

Наименование	Acer Aspire V5 (ЭВМ1)	Asus K56CB (ЭВМ2)	Raspberry Pi 3 B (ЭВМ3)
Процессор	Intel i7-6700HQ	Intel i5-3317U	Broadcom 2837
Тактовая частота	2,6 GHz x 4	1,7 GHz x 2	1,2GHz x 4
Объем ОЗУ	16 GB	8 GB	1 GB
Операционная система	Linux Mint 18.1	Linux Mint 18.1	Raspbian Linux
Версия ОС	amd64	amd64	Armv71
Компилятор языка Си	GCC 5.4.0	GCC 5.4.0	GCC 4.9.2

На рис. 6 приведены графики зависимости времени выполнения 10^8 операций умножения от степени поля Галуа GF(2^m). График времени выполнения алгоритма Карацубы не приведен, так как время его выполнения на несколько порядков больше времени выполнения рассмотренных алгоритмов. Для сравнения на графике приведено время выполнения «пустой» программы, не содержащей команд операции умножения.

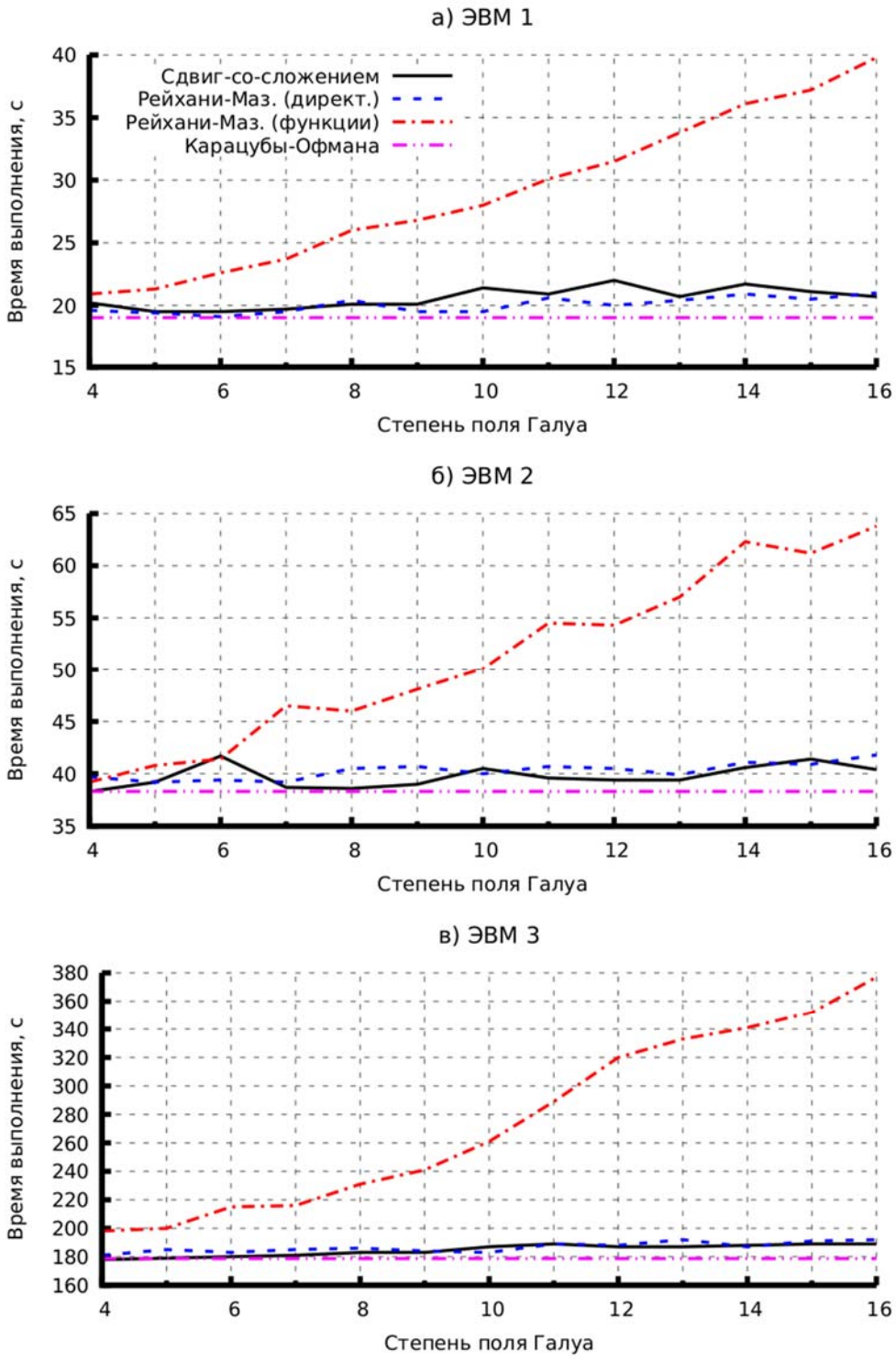

Рис. 6. Графики зависимости времени выполнения 10^8 операций умножения от степени поля Галуа GF(2^m) для различных умножителей

Из графиков видно, что наилучшее быстродействие из рассмотренных алгоритмов показывают алгоритм умножения Рейхани-Мазолеха, выполненный с помощью директив define, и алгоритм «сдвиг-со-сложением, справа-налево». При этом, для некоторых степеней поля Галуа алгоритм Рейхани-Мазолеха показывает несколько меньшее время выполнения.

Быстродействие алгоритма Рейхани-Мазолеха зависит от нескольких факторов: степени поля Галуа, вида порождающего полинома и конкретной программной реализации алгоритма. Чем выше степень образующего полинома, тем больше размерность матриц, на основе которых производятся вычисления в этом алгоритме, а увеличение размерности влечет за собой увеличение количества операций, необходимых для получения результата. Такие матрицы имеют различную сложность в зависимости от образующего полинома.

На скорость выполнения умножения в значительной степени влияет конкретная реализация. В случае реализации алгоритма при помощи функций появляются накладные расходы на итерирование по массивам, проверку условий и вызов вспомогательных функций.

Помимо прочего, свой вклад вносят и характеристики ЭВМ, такие как объем оперативной памяти, частота и архитектура процессора, используемая операционная система. На графиках видно, что скорость выполнения алгоритма на процессоре с архитектурой arm (ЭВМ 3) намного ниже, чем на amd64, это связано с более низкой тактовой частотой и особенностью структуры памяти.

Необходимо отметить, что построение умножителя Рейхани-Мазолеха с использованием функций проще с точки зрения написания программного кода. Однако, использование программы-генератора как раз и позволяет разработчику использовать более сложные по реализации, но быстрые версии алгоритмов, без дополнительных затрат на написание программного кода. Предполагается использование сформированного программного кода для создания прошивок специализированных встраиваемых в оборудование микроконтроллеров. При этом оценивать их быстродействие должны сами разработчики, исходя из технических требований к разрабатываемой системе передачи.

Таким образом, при разработке систем передачи данных, использующих вычисления в конечных полях, можно рекомендовать использовать алгоритм «сдвиг-со-сложением, справа-налево» и алгоритм умножения Рейхани-Мазолеха, выполненный с помощью директив define, которые обеспечивают более высокое быстродействие в сравнении с другими рассмотренными алгоритмами.

Заключение

В статье был рассмотрен реализованный авторами программный комплекс для автоматического построения умножителей элементов двоичного поля Галуа, решающий одну из задач автоматизации процесса разработки систем кодирования и шифрования, основанных на операциях в конечных полях. Также проведено сравнение реализованных в программе-генераторе алгоритмов с точки зрения быстродействия и даны рекомендации по выбору умножителя элементов поля при построении систем передачи данных.

Литература

1. Баричев С. Г., Гончаров В. В., Серов Р. Е. Основы современной криптографии. М.: Горячая Линия – Телеком, 2011. 175 с. ISBN 978-5-9912-0182-7.

2. Robert H. Morelos-Zaragoza The Art of Error Correcting Coding. Chichester: John Wiley & Sons, Ltd, 2002. 232 p. ISBN 0471-49581-6.

3. Когновицкий О. С. Двойственный базис и его применение в телекоммуникациях. СПб.: Линк, 2009. 423 с.

4. Кукунин Д. С. Анализ эффективности декодирования циклических кодов с использованием двойственного базиса: диссертация. СПб.: СПбГУТ, 2009. 197 с.

5. Владимиров С. С. Анализ эффективности декодирования циклических кодов Рида-Соломона с использованием двойственного базиса: диссертация. СПб.: СПбГУТ, 2013. 159 с.

6. Карацуба А., Офман Ю. Умножение многозначных чисел на автоматах // Доклады Академии Наук СССР. 1962. Т. 145. С. 293–294.

7. Rodriguez-Henriquez F., Koc C. K. On fully parallel Karatsuba Multipliers for GF (2m) // International Conference on Computer Science and Technology (CST). 2003. pp. 405–410.

8. Владимиров С. С. Математические основы теории помехоустойчивого кодирования. СПб.: СПбГУТ, 2016. 96 с. ISBN 978-5-89160-131-4.

9. Reyhani-Masoleh A., Hasan M. A. Low complexity bit parallel architectures for polynomial basis multiplication over GF(2m) // IEEE Transactions on Computers. 2004. Vol. 53. No. 8. pp. 945–959.

10. Владимиров С. С. Эффективность умножителя Рейхани-Мазолеха элементов двоичного поля Галуа // Информационные технологии и телекоммуникации. 2015. № 3 (11). С. 84–92. URL: http://www.sut.ru/doci/nauka/review/3-15.pdf

11. Владимиров С. С., Мухаметшина Д. Ф. Разработка программного комплекса для автоматизации построения умножителя элементов двоичного поля Галуа по схеме Рейхани-Мазолеха // Информационные технологии и телекоммуникации. 2017. Том 5. № 1. С. 68–77. URL: http://www.sut.ru/doci/ nauka/ review/ 20171/68-77.pdf

References

1. Barichev S., Goncharov V., Serov R. Fundamentals of Modern Cryptography. M.: Goryachaya liniya – Telecom. 2011. 175 p.

2. Robert H. Morelos-Zaragoza The Art of Error Correcting Coding. Chichester: John Wiley & Sons, Ltd. 2002. 232 p. ISBN 0471-49581-6.

3. Kognovitsky, O. Dual Basis and its Application in Telecommunications. SPb.: Link. 2009. 423 p.

4. Kukunin D. Analysis of Decoding Efficiency of Cyclic Codes using a Dual Basis: Dissertation. SPb.: SPbGUT. 2009. 197 p.

5. Vladimirov S. Analysis of Decoding Efficiency of Cyclic Reed-Solomon Codes using a Dual Basis: Dissertation. SPb.: SPbGUT. 2013. 159 p.

6. Karatsuba A., Ofman Y. Multiplication of Multidigit Numbers on Automata // The Proceedings of the USSR Academy of Sciences. 1962. Vol. 145. pp. 293–294.

7. Rodriguez-Henriquez F., Koc C. K. On fully parallel Karatsuba Multipliers for GF (2m) // International Conference on Computer Science and Technology (CST). 2003. pp. 405–410.

8. Vladimirov, S. Mathematical Foundations of the theory of Error Correcting Coding. SPb.: SPbGUT. 2016. 96 p. ISBN 978-5-89160-131-4.

9. Reyhani-Masoleh A., Hasan M. A. Low complexity bit parallel architectures for polynomial basis multiplication over GF(2m) // IEEE Transactions on Computers. 2004. Vol. 53, no. 8. pp. 945–959.

10. Vladimirov S. The Efficiency of Binary Galois Field Elements Reyhani-Masoleh Multiplier // Telecom IT. 2015. Vol. 3 (11). pp. 84–92. URL: http://www.sut.ru/doci/nauka/review/3-15.pdf

11. Vladimirov S., Mukhametshina D.: The Research of the Software for Automatic Developing of Reyhani-Masoleh Multiplier over Galois Field // Telecom IT. 2017. Vol. 5. Iss. 1. pp. 68–77 (in Russian). URL: http://www.sut.ru/doci/nauka/ review/20171/68-77.pdf

Мухаметшина Дина Фаиловна – магистр, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, undinakamec@gmail.com

Владимиров Сергей Сергеевич – кандидат технических наук, ассистент, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, vladimirov.opds@gmail.com

Mukhametshina Dina – Undergraduate, SPbSUT, St. Petersburg, 193232, Russian Federation, undinakamec@gmail.com

Vladimirov Sergey – Candidate of Engineering Sciences, Assistant, SPbSUT, St. Petersburg, 193232, Russian Federation, vladimirov.opds@gmail.com