УДК 004.272.2

Полиномиальный модулярный умножитель в устройствах помехоустойчивого кодирования

П.С. Поперечный

Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН),

Научно-производственный центр «ЭЛВИС», pporerechny@elvees.com

***Аннотация* — В помехоустойчивом кодировании основной операцией является умножение, выполняемое в расширенном поле Галуа. При увеличении расширения поля растет сложность умножителей. В статье предложен подход к использованию принципов системы остаточных классов в применении к полиномиальной арифметике. Данный подход увеличивает быстродействие за счет распараллеливания операции умножения по нескольким простым модулям. Прямые и обратные преобразователи просчитываются единожды перед логическим синтезом аппаратной реализации кодеков.**

***Ключевые слова* — помехоустойчивое кодирование, расширенное поле Галуа, примитивный многочлен, полиномиальный умножитель, система остаточных классов.**

I. Введение. Постанова задачи

Помехоустойчивое кодирование нашло широкое применение в системах приема-передачи и хранения информации. Для кодов БЧХ (Боузе-Чоудхурри-Хоквингема) и РС (Рида-Соломона) разработано множество алгоритмов декодирования с адаптацией под современную элементную базу [1]. В алгоритмах кодирования-декодирования используются разные арифметические операции (сумма, произведение, возведение в степень, формальная производная, деление), но все они сводятся к операциям суммирования по модулю 2 и умножению по модулю примитивного многочлена в расширенном поле Галуа [2]. Умножение двух векторов в расширенном поле $GF(2^m)$, заданном с помощью неприводимого примитивного многочлена $p(x)$ степени m, осуществляется путем вычисления суммы результатов умножения одного вектора на каждый член другого вектора; результат приводится по модулю $p(x)$ [1]:

$$A(x)\cdot B(x)=\left(\sum_{i=0}^{m-1}A_i\cdot x^i\cdot\left(\sum_{j=0}^{m-1}B_j\cdot x^j\right)\right)\bmod p(x)\ ,$$

где $A_i, B_j \in GF(2)$.

Выбранная степень поля m зависит от длины кодируемых данных. При увеличении размерности поля возрастает сложность умножителей, и, как следствие, нелинейно увеличиваются площадь аппаратной реализации и временные задержки вычислительного модуля.

В статье предложен подход к использованию системы остаточных классов (СОК) для распараллеливания операции полиномиального умножения с целью увеличения быстродействия. Также данный подход может обеспечить универсальность умножителя для любого расширения поля.

Система остаточных классов (модулярная арифметика) определяется набором N целых, попарно взаимно простых чисел $\{p_1, p_2, ..., p_N\}$, которые принято называть модулями. Произведение этих чисел определяет динамический диапазон $M = p_1 \cdot p_2 \cdot .... \cdot p_N$ системы. Это означает, что любое число A в пределах диапазона имеет уникальное представление в модулярной арифметике $\{a_i, a_2, ..., a_N\}$, где $a_i = |A|_{pi}$. При условии, что результат выполнения операции не выходит за выбранный диапазон, арифметические операции сложения, вычитания и умножения выполняются над вычетами параллельно, т.е. независимо по каждому модульному основанию [3, 4]:

$$|A+B|_M = \{|a_1+b_1|_{p1}, |a_2+b_2|_{p2}, ..., |a_N+b_N|_{pN}\}\ ,$$

$$|A\cdot B|_M = \{|a_1\cdot b_1|_{p1}, |a_2\cdot b_2|_{p2}, ..., |a_N\cdot b_N|_{pN}\}\ .$$

Здесь и далее по тексту прямые скобки означают взятие по соответствующему модулю.

II. Применение системы остаточных классов в полиномиальной арифметике

Для применения системы остаточных классов в полиномиальной арифметике нужно учитывать некоторые особенности:

- вместо попарно простых чисел $\{p_1, p_2, ..., p_N\}$ необходимо выбрать примитивные неприводимые полиномы $\{p_1(x), p_2(x), ..., p_N(x)\}$ небольшой степени [2];

- степень динамического диапазона должна быть выбрана из неравенства: $\deg M(x) = \deg\left(p_1(x)\cdot p_2(x)\cdot ....\cdot p_N(x)\right) > > \deg(A(x)\cdot B(x))$;
- для модулярного представления вектора его необходимо привести по модулям примитивных полиномов: $\left|A(x)\right|_{M(x)} = \{\left|a_1(x)\right|_{p1(x)}, \left|a_2(x)\right|_{p2(x)}, ..., \left|a_N(x)\right|_{pN(x)}\}$;
- умножение векторов в модулярном виде: $\left|A(x)\cdot B(x)\right|_{M(x)} = \{\left|a_1(x)\cdot b_1(x)\right|_{p1(x)}, ..., \left|a_N(x)\cdot b_N(x)\right|_{pN(x)}\}$ осуществляется по модулям простых полиномов малых степеней;
- обратное преобразование можно выполнить через полиадический код;
- результат $\left|A(x)\cdot B(x)\right|_{M(x)}$ необходимо привести к модулю первоначального примитивного полинома $p(x)$, задающему расширенное поле $GF(2^m)$.

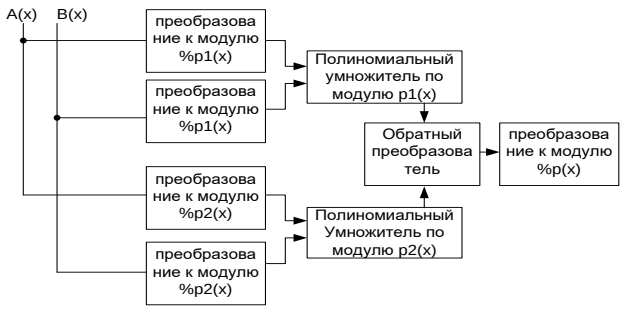

Рис. 1. Структурная схема полиномиального умножителя в СОК

Для приведения векторов к модулям примитивных полиномов $\{p_1(x), p_2(x), ..., p_N(x)\}$, а также к модулю задающего примитивного полинома $p(x)$, в схемотехнике применяется регистр с линейной обратной связью (РЛОС), выполняющий деление на заданный полином [1]. Данная схема позволяет получить остаток от деления полинома, компоненты которого поступают последовательно. Для однотактного выполнения деления на полином можно применить схему распараллеливания РЛОС, представленную в [5, 6], задав вместо порождающего полинома $g(x)$ один из примитивных полиномов $\{p_1(x), p_2(x), ..., p_N(x)\}$.

III. Обратное преобразование

Степень динамического диапазона можно определить из выражения:

$$M(x) = p_1(x)\cdot p_2(x)\cdot ....\cdot p_N(x). \quad (1)$$

Вспомогательные вектора:

$$\begin{matrix} M1(x) = M(x)/\,p1(x) \\ \vdots \\ MN(x) = M(x)/\,pN(x) \end{matrix}. \quad (2)$$

Должны выполняться следующие условия [4]:

$$\begin{matrix} \left|M1(x)\cdot b1(x)\right|_{p1(x)} = 1 \\ \vdots \\ \left|MN(x)\cdot bN(x)\right|_{pN(x)} = 1 \end{matrix}. \quad (3)$$

Тогда для обратного преобразования из модулярного представления в первоначальное можно воспользоваться следующим выражением:

$$A(x) = \left|\left|a_1(x)\cdot M1(x)\cdot b1(x) + .. + a_N(x)\cdot MN(x)\cdot bN(x)\right|_{M(x)}\right|_{p(x)}. \quad (4)$$

IV. Численный пример умножения двух векторов

Для наглядного представления описываемого способа в таблице 1 построим поле $GF(2^4)$ над примитивным многочленом $p(x) = x^4 + x + 1$ [1]:

Таблица 1

Компоненты поля $GF(2^4)$ с примитивным многочленом $p(x) = x^4 + x + 1$

Степенное представление	Полиномиальное представление	Векторное представление
0	0	(0 0 0 0)
1	1	(0 0 0 1)
c	c	(0 0 1 0)
c^2	c^2	(0 1 0 0)
c^3	c^3	(1 0 0 0)
c^4	$c + 1$	(0 0 1 1)
c^5	$c^2 + c$	(0 1 1 0)
c^6	$c^3 + c^2$	(1 1 0 0)
c^7	$c^3 + c + 1$	(1 0 1 1)
c^8	$c^2 + 1$	(0 1 0 1)
c^9	$c^3 + c$	(1 0 1 0)
c^{10}	$c^2 + c + 1$	(0 1 1 1)
c^{11}	$c^3 + c^2 + c$	(1 1 1 0)
c^{12}	$c^3 + c^2 + c + 1$	(1 1 1 1)
c^{13}	$c^3 + c^2 + 1$	(1 1 0 1)
c^{14}	$c^3 + 1$	(1 0 0 1)
$c^{15} = 1$	1	(0 0 0 1)

Выберем простые примитивные полиномы меньшего порядка, служащие модулями для системы остаточных классов. Выберем два полинома $p1(x) = x^3 + x + 1$, $p2(x) = x^2 + x + 1$ из

таблицы примитивных неприводимых полиномов [2]. Тогда, согласно (1), динамический диапазон равен:

$$M(x) = p1(x) \cdot p2(x) = (x^3 + x + 1) \cdot (x^2 + x + 1) =$$
$$x^5 + x^4 + x^3 + x^3 + x^2 + x + x^2 + x + 1 = x^5 + x^4 + 1.$$

Все вектора и полиномы хоть и принадлежат разным порядкам поля, однако все одного типа $GF(2^m)$, поэтому сложение коэффициентов при одинаковых степенях выполняется по модулю 2.

Для обратного преобразования найдем следующие вспомогательные вектора по формулам (2):

$$M1(x) = M(x) / p1(x) = p2(x),$$

$$M2(x) = M(x) / p2(x) = p1(x).$$

Также должны выполняться следующие условия (3):

$$\left|M1(x) \cdot b1(x)\right|_{p1(x)} = 1 ,$$

$$\left|M2(x) \cdot b2(x)\right|_{p2(x)} = 1.$$

Решив эти уравнения можно найти:

$$b1(x) = x^2 \ , \ b2(x) = x + 1.$$

Сделав предварительные вычисления, можно перейти непосредственно к модульной арифметике.

Выполним сложение двух элементов из таблицы 1:

$$A(x) + B(x) = c^7 + c^5 = (c^3 + c + 1) + (c^2 + c) = c^3 + c^2 + 1. \quad (5)$$

Очевидно, что для операции сложения нецелесообразно использовать предложенный подход, однако в общем случае операции сложения и умножения можно выполнять таким образом. Проверим результат, полученный в (5), при вычислении в модульном виде.

$$A(x) = c^3 + c + 1 = \{\left|a_1(x)\right|_{p1(x)}, \left|a_2(x)\right|_{pN(x)}\} = \{0, c\},$$

$$B(x) = c^2 + c = \{\left|b_1(x)\right|_{p1(x)}, \left|b_2(x)\right|_{pN(x)}\} = \{c^2 + c, 1\},$$

тогда $A(x) + B(x) = \{c^2 + c,\ c + 1\}$.

Для обратного преобразования полученного результата применим формулу (4):

$$A(x) + B(x) =$$
$$= \left|(c^2 + c) \cdot (c^2 + c + 1) \cdot c^2 + (c + 1) \cdot (c^3 + c + 1) \cdot (c + 1)\right|_{p(x)} =$$
$$= \left|\left(c^6 + c^3\right) + (c^5 + c^2 + c + 1)\right|_{p(x)} = c^3 + c^2 + 1.$$

Результат сходится с результатом, полученным в (5). Выполним умножение выбранных элементов:

$$A(x) \cdot B(x) = \left|c^7 \cdot c^5\right|_{p(x)} = \left|c^{12}\right|_{p(x)} = c^3 + c^2 + c + 1. \quad (6)$$

Выполним то же умножение, но в модульном виде:

$$A(x) \cdot B(x) = \{0 \cdot (c^2 + c),\ c \cdot 1\} = \{0,\ c\} \ .$$

Видим, что результат в модульном виде сходится с модульным представлением вектора $A(x) = \{0,\ c\}$.

Неувязка заключается в том, что произведение векторов без приведения к модулю $p(x)$ выходит за пределы динамического диапазона $M(x)$ (см. выше). Таким образом, динамический диапазон для умножения всех векторов из таблицы 1 был выбран недостаточным. Однако это не мешает выполнить произведение векторов меньшего порядка, например:

$$D(x) \cdot A(x) = c^4 \cdot c^7 = c^{11} = c^3 + c^2 + c \ . \quad (7)$$

В модульном виде:

$$D(x) = c^4 = c + 1 = \{\left|c_1(x)\right|_{p1(x)}, \left|c_2(x)\right|_{p2(x)}\} = \{c + 1,\ c + 1\},$$

тогда $D(x) \cdot A(x) = \{(c + 1) \cdot 0,\ (c + 1) \cdot c\} = \{0,\ c^2 + c\}$.

Выполним обратное преобразование (4):

$$D(x) \cdot A(x) =$$
$$= \left|\left|0 \cdot (c^2 + c + 1) \cdot c^2 + (c^2 + c) \cdot (c^3 + c + 1) \cdot (c + 1)\right|_{M(x)}\right|_{p(x)} =$$
$$= \left|\left|c^6 + c^3 + c^2 + c\right|_{M(x)}\right|_{p(x)} = \left|c^4 + c^3 + c^2 + 1\right|_{p(x)} = c^3 + c^2 + c.$$

Результат сходится с результатом (7).

V. Заключение

Устройства помехоустойчивого кодирования (кодер-декодер) используются в различных применениях, например, для надежного хранения данных в различных типах носителей информации, в системах коммуникации и т.д. Поэтому выбранная длина данных не фиксирована в конкретном случае, а значит существует необходимость использования кодеков, рассчитанных под разное расширение поля Галуа. Описанная сложность в основном упирается в реализацию умножителя не в фиксированном поле.

Предложенный подход в реализации умножителя позволяет «покинуть» заданное поле прямым преобразованием входных векторов к модульным представлениям, реализуемым простым делением на полином [5, 6]. При этом необходимо лишь выбрать достаточный динамический диапазон, а далее вся арифметика происходит по нескольким простым модулям (примитивным полиномам). Основные принципы СОК применимы и в полиномиальной арифметике, вместо взаимнопростых модулей выбираются примитивные неприводимые многочлены.

Реализованный умножитель обладает следующими преимуществами:

- быстродействие операции умножения может быть увеличено при достаточно больших порождающих полиномах за счет распараллеливания на полиномиальные умножители меньших порядков, которые для

еще большего ускорения можно реализовать таблично (при этом ввиду их малого порядка размер таблицы будет крайне малым) [7, 8];

- прямое и обратное преобразование к модулям осуществляется простым делением входных полиномов на неизменяемые в процессе работы примитивные полиномы (деление на постоянный полином можно выполнить параллельным РЛОС [5, 6]);
- латентность схемы несколько возрастает за счет использования прямых и обратных преобразователей, однако данную схему можно конвейеризовать ввиду независимости этапов вычислений.

Литература

[1] Блейхут Р. Теория и практика кодов, контролирующих ошибки. Пер. с англ. М.: Мир, 1986, 576 с.

[2] Рахман П.А. Кодирование информации с применением кодов Рида-Соломона URL: http://bugtraq.ru/library/crypto/.keep/rscodes.pdf (дата обращеня 01.02.2016).

[3] Амербаев В.М., Соловьев Р.А.,Тельпухов Д.В., Поперечный П.С.,Рухлов В.С.,Щелоков А.Н.,Михмель А.С.. Разработка устройства для вычисления результата операции скалярного произведения векторов на базе интрамодулярного разложения комплексных чисел в модулярной арифметике // Известия ЮФУ. Технические науки. -2015. -№6. -С.95-105.

[4] Omondi A., Premkuar B. Residue Number System: Theory and Implementation // Imperial College Press. – 2007. ISBN 978-1-86094-866-4.

[5] Патент RU157943. Параллельный реконфигурируемый кодер БЧХ кодов. 21.07.2015.

[6] Поперечный П.С. Разработка параллельного кодера БЧХ с регулируемой корректирующей способностью // Известия ЮФУ. Технические науки. -2015. -№7. -с.19-31.

[7] Белецкий А.Я., Белецкий Е.А., Воливач О.И., Якимчук М.А. Синтез и анализ кодов Рида-Соломона в пространстве изоморфного изобажения // Інформаційні технології в освіті. Национальный авиационный университет, Киев – 2013. - №17. – с.38-55.

[8] Амербаев В.М., Балака Е.С., Соловьев Р.А., Тельпухов Д.В. Анализ и синтез арифметического узла проф. Поспелова Д.А. поля Галуа // VI Всероссийская научно-техническая конференция МЭС-2014. Сборник трудов. Часть IV.-М.: ИППМ РАН, 2014.- с.179-182.

Polynomial modular multipliers for error correcting code devices

P.S. Poperechny

Postgraduate of Institute for Design Problems in Microelectronics RAS,

Engeneer of Research and Development Center "ELVEES", ppoperechny@elvees.com

***Keywords* — Error-correction coding, extended Galois Field, primitive polynomial, polynomial multiplier, residue number system, SoC.**

Abstract

It is known that multiplication is widespread and complicated arithmetic operation in error correction coding. In spite of being formal derivative, division and rising to power operations in RS-codes, all of these operations can be collapsed to adding and multiplying in extended Galois Field. Once data size is chosen, all vector multiplying is performed by the same primitive polynomial that generates the Field. Therefore, in case of large Field polynomial multipliers become very complicated and critical path length increases nonlinearly. As input vectors are represented by residue polynomials, all calculations are shared by a small parallel calculating block. Such blocks have small critical path delay, so common performance throughput is increased.

References

[1] Blejhut R. Theory and practice of error-controlling codes. Moscow, Mir, 1986, 576 p. (in Russian)

[2] Rahman P.A. Data coding by Reed-Solomon code URL: http://bugtraq.ru/library/crypto/.keep/rscodes.pdf (01.02.2016) (in Russian).

[3] Amerbaev V.M., Solov'ev R.A.,Tel'puhov D.V., Poperechnyj P.S.,Ruhlov V.S.,Shhelokov A.N.,Mihmel' A.S.. Design of device for scalar multiplication result calculation of vectors by means of intromodulated decomposition of complex numbers for residue number system. Izvestija JuFU. Tehnicheskie nauki. 2015. No.6. pp.95-105 (in Russian).

[4] Omondi A., Premkuar B. Residue Number System: Theory and Implementation. Imperial College Press. 2007. ISBN 978-1-86094-866-4.

[5] Patent RU157943. Parallel reconfigurable BCH encoder. 21.07.2015 (in Russian).

[6] Poperechnyj P.S. Design of parallel BCH encoder with tunable correcting level. Izvestija JuFU. Tehnicheskie nauki. 2015. No.7. pp.19-31 (in Russian).

[7] Beleckij A.Ja., Beleckij E.A., Volivach O.I., Jakimchuk M.A. Synthesis and Analysis RS codes at isomorphic image. Informacijni tehnologiï v osviti. Nacional'nyj aviacionnyj universitet, Kiev. 2013. No.17. pp.38-55 (in Russian).

[8] Amerbaev V.M., Balaka E.S., Solov'ev R.A., Tel'puhov D.V. Analysis and Synthesis arithmetic node of prof. Pospelov in Galois Field. VI Vserossijskaja nauchno-tehnicheskaja konferencija MJeS-2014. Sbornik trudov. Chast' IV.-Moscow, IPPM RAN, 2014. pp.179-182 (in Russian).

.