ПРОЕКТИРОВАНИЕ КОДЕРА И ДЕКОДЕРА КОДА РИДА-СОЛОМОНА RS (255, 251) В FPGA

Anindya Sundar Das, Satyajit Das and Jaydeb Bhaumik

International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013

Автор перевода: Дяченко В.О.

Реферат - Обнаружение и исправление ошибок в цифровых данных является важной проблемой для современных систем связи. Поэтому для защиты цифровых данных необходим эффективный код исправления ошибок. В высокоскоростных системах связи коды Рида-Соломона широко используются для обеспечения защиты от ошибок, особенно против пакетных ошибок. Коды Рида-Соломона являются циклическими, не двоичными кодами. В этой статье кодер и декодер кода Рида-Соломона RS (255, 251) были разработаны и реализованы на платформе FPGA.

Ключевые слова - коды Рида-Соломона, поле Галуа, кодировщик RS, декодер RS.

ВВЕДЕНИЕ

Коды коррекции ошибок имеют широкий спектр приложений в различных областях, таких как цифровая передача данных, проектирование системы памяти и отказоустойчивых компьютерных систем. Код Рида-Соломона (RS) является хорошо известным недвоичным линейным блочным кодом. Он широко используется для исправления ошибок во многих приложениях, таких как устройства хранения (CD, DVD), беспроводная связь, высокоскоростные модемы и спутниковая связь. Например, коды RS (28, 24) и RS (32, 28) с чередованием широко используются для хранения данных на компакт-дисках.

Для цифрового микроволнового радио использовался 16-битный код RS (255, 223) с исправлением ошибок. Код RS (255, 239) с возможностью исправления 8 байтов ошибок был рекомендован в качестве внешнего кода в WiMax. Для сохранения важной информации заголовка MB-OFDM UWB, используются коды RS (23, 17).

Основы RS-кодов и ряд методов декодирования можно найти в литературе [1] [2]. Несколько важных применений кодов RS обсуждались в [3]. Параллельная архитектура высокоскоростного кодека RS (255, 251) была предложена в [4]. Высокоскоростная архитектура для декодеров Рида-Соломона предлагается в [5]. Сложность кодера и декодера RS возрастает с исправляющими возможностями кода. Следовательно, многие исследователи направили свои усилия на минимизацию сложности RS-декодера. В конструкции системы СБИС энергопотребление вызывает серьезную озабоченность, и в то же время площадь кремния должна поддерживаться как

можно ниже. В этом случае наша цель заключается в разработке кодирующего и декодирующего устройств на основе FPGA (255, 251), имеющего возможности исправления ошибок в двух байтах.

В следующем разделе дается краткое введение в теорию коодов Рида-Соломона. В разделе III был описан кодер кода RS и соответствующая ему архитектура. Основные блоки декодера RS и соответствующие схемы описаны в разделе IV. Результат синтеза кодека RS (255, 251) представлен в разделе V, и, наконец, статья завершается в разделе VI.

I. КОДЫ РИДА-СОЛОМОНА

Коды Рида-Соломона представляют собой линейные блоковые коды и подмножеством кодов БЧХ. Код RS основан на конечных полях, часто называемых полями Галуа. Количество и типы ошибок, которые могут быть исправлены, зависят от характеристик кода Рида-Соломона. Параметры RS (n, k) кодов описываются следующим образом.

m - количество бит на символ, k - длина незашифрованного сообщения в символах, n - длина кодового слова в символах (n-k) - количество символов проверки четности, t - количество исправляемых ошибок символов, 2t = n-k.

На рисунке 1 показано типичное систематическое кодовое слово Рида-Соломона. Это систематический код, поскольку данные слева неизменны, а символы четности добавляются вместе с данными для формирования кодового слова.

Рис.1: Кодовое слово кода Рида-Соломона RS (n, k)

Код RS с 8-битными символами будет использовать поле Галуа GF (2^8). Для кода RS (255,251) n = длина блока = 255, k = количеству информационных символов = 251, 2t = (n-k) = количество символов четности = 4, t = максимальное количество ошибок, которое может быть исправлено = 2. Исходное сообщение можно восстановить, используя RS-декодер, при условии, что количество ошибок в принятом кодовом слове меньше или равно двум.

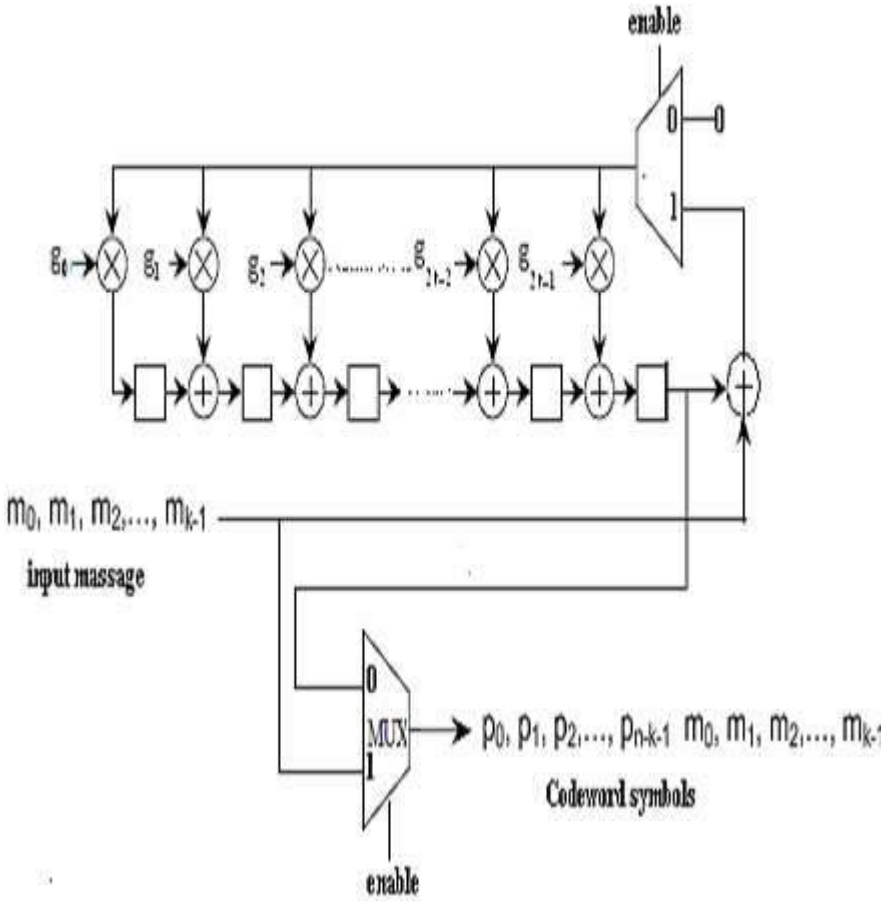
III. RS ENCODER

Кодер RS принимает блок символов и добавляет дополнительные символы проверки четности. Кодовое слово RS может быть вычислено из символов входных сообщений с использованием генераторного многочлена. Многочлен генератора зависит от порядка поля Галуа, над которым был определен код RS, и числа ошибок, которые нужно исправить. Примитивный многочлен определяется порядком поля Галуа. Одним из примитивных полиномов в поле GF (2^8) является $x^8 + x^4 + x^3 + x^2 + 1$. Пусть α - примитивный элемент в поле Галуа в GF (2^8). Тогда генераторный многочлен для t-кода с исправлением ошибок RS выглядит следующим образом.

$$g(x) = \prod_{i=1}^{2t}(x + \alpha^i) \qquad (1)$$

Для двоичного байтового кода с исправлением ошибок (t = 2) соответствующий генераторный многочлен

$$g(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^3)(x + \alpha^4) \quad (2)$$



Пусть входной многочлен M(x) кодера

$$M(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \cdots + m_1x + m_0 \quad (3)$$

Соответствующий многочлен кодового слова задается формулой

$$C(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_1x + c_0 \quad (4)$$

Это кодовое слово генерируется из входного информационного полинома, используя следующую формулу.

$$C(x) = x^{2t}M(x) + x^{2t}M(x) \bmod g(x) \quad (5)$$

Если Q (x) и P (x) - соответствующие полиномы частного и остатка, когда $x^{2t}M(x)$ делится на g (x), то многочлен кодового слова также может быть выражен другим способом.

$$C(x) = x^{2t}M(x) + P(x) = Q(x)g(x) \quad (6)$$

Здесь P(x) - полином проверки четности. Из уравнения (6) можно получить

$$\frac{x^{(n-k)}M(x)}{g(x)} = Q(x) - \frac{P(x)}{g(x)} \quad (7)$$

Добавляя полином P(x) проверки на четность к полиному x^{2t} M (x), вычисляется полином C(x) кодового слова, который нацело делится на g(x).

IV. ДЕКОДЕР КОДА РИДА-СОЛОМОНА

Декодер обрабатывает каждый блок и пытается исправить ошибки, которые могут возникнуть во время передачи или хранения. Декодер делит полученный полином на кодовый генераторный полином кода RS. Если остаток равен нулю, то ошибок не обнаружено, в противном случае - ошибки были.

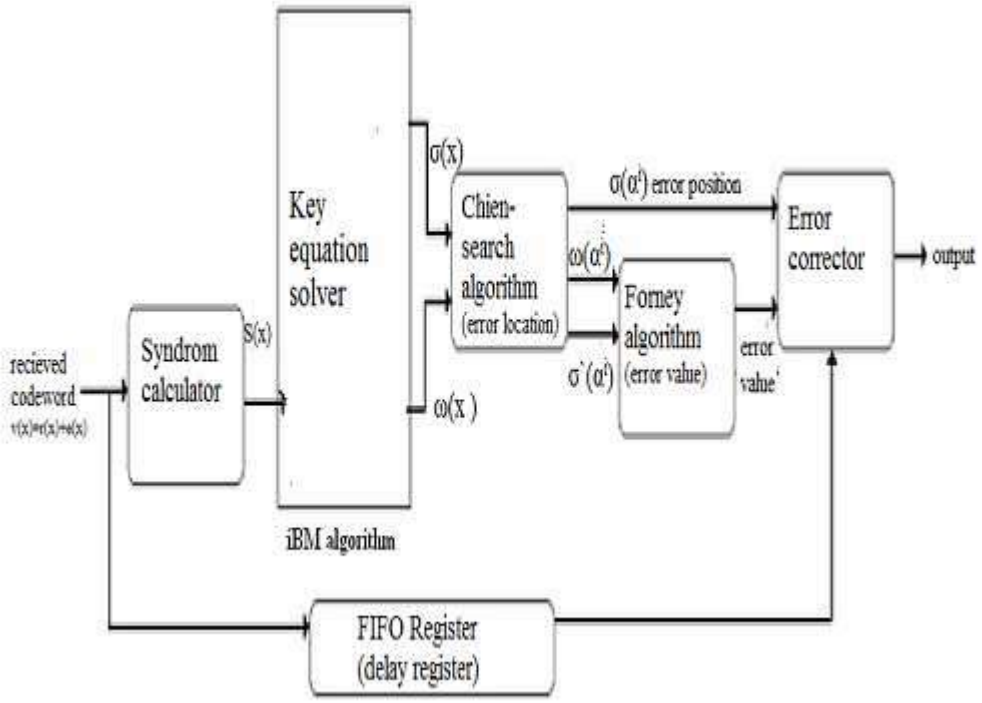


Рис. 3: Блок-схема декодера Рида-Соломона

Типичный RS-декодер имеет пять этапов в цикле декодирования, а именно:

1. Формирователь синдрома

2. Решатель ключевых уравнений - вычисление значений ошибок и их соответствующих местоположений.

3. Поиск Ченя - решение полинома локатора ошибок

4. Алгоритм Форни для вычисления величины ошибки

5. Коррекция ошибок

A. Генератор синдрома

Генерация синдрома из принятого кодового слова является первым этапом процесса декодирования. Здесь вычисляются синдромы и определяется, есть ли ошибки в полученном кодовом слове или нет.

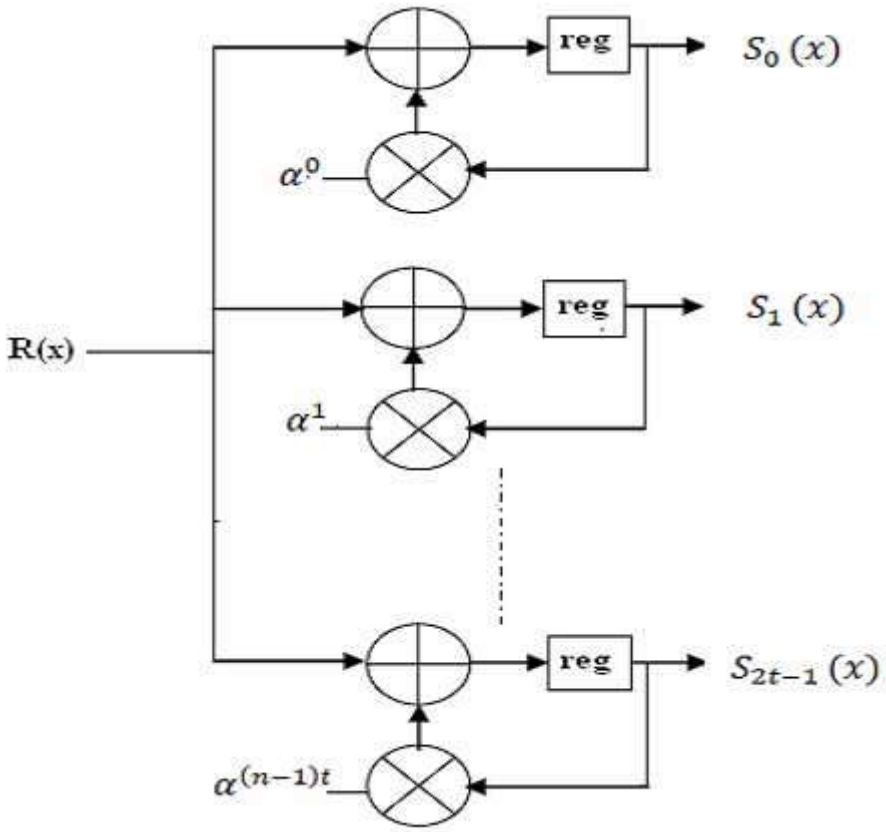


Рис. 4. Архитектура генератора синдрома

Полином синдрома обычно представлен как

$$S(x) = S_0 + S_1 x + S_2 x^2 + \cdots + S_{(2t-1)} x^{(2t-1)} = \sum_{i=0}^{(2t-1)} S_i x^{(i-1)} \quad (8)$$

Предположим, что R (x) - принятый многочлен кодового слова

$$R(x) = R_0 + R_1(x) + R_2 x^2 + \cdots + R_{n-1} x^{n-1} \quad (9)$$

Тогда i-й синдром может быть выражен следующим образом

$$S_i = R_0 + \alpha^i(R_1 + \alpha^i(R_2 + \alpha^i(R_3 + \cdots + \alpha^i(R_{n-1}) \ldots))) \quad (10)$$

,

где R_{n-1} является первым принятым символом.

B. Решатель ключевых уравнений

Следующим шагом после вычисления полинома синдрома является вычисление значений ошибок и их соответствующих местоположений. Этот этап включает в себя решение многочленов синдрома 2t, сформированных на предыдущем этапе. Эти многочлены имеют ν неизвестных, где ν - количество ошибок, возникающих в полученном кодовом слове. Если неизвестные местоположения i_1, i_2 ... i_ν, то полином ошибки может быть выражен как

$$E(x) = Y_1 x^{i_1} + Y_2 x^{i_2} + \cdots + Y_\nu x^{i_\nu} \qquad (11)$$

где Y_r - величина r-й ошибки в местоположении i^r. Если x_r - элемент поля, связанный с местоположением ошибки i_r, то коэффициенты синдрома определяются выражением

$$S_j = \sum_{t=1}^{\nu} Y_r x_r^t \qquad (12)$$

,

где j = 1,2 ... 2t. Y_r - значение ошибки, а X_r - местоположение ошибки r-го символа ошибки. Многочлен локатора ошибок можно определить как

$$\sigma(x) = \sigma_\nu x^\nu + \sigma_{\nu-1} x^{\nu-1} + \cdots + \sigma_1 x + \sigma_0 \qquad (13)$$

Решение для коэффициентов σ(x) представляет полиномиальный локатор ошибки. Этот полином σ(x) связан с полиномом значения ошибки с помощью некоторого уравнения, которое называется ключевым уравнением. Если значение ошибки определяется формулой ω(x), ключевое уравнение выражается как

$$S(x)\sigma(x) = \omega(x) \bmod x^{2t} \qquad (14)$$

Тогда ω(x) можно использовать для определения величин ошибок

Y_1, ... Y_ν.

Эти уравнения решаются по-разному. В этой работе мы используем алгоритм «инверсный алгоритм Берлекэмпа-Месси» (алгоритм iBM). Алгоритм описан ниже: [5]

Initialization

$\Lambda_0(0) = b_0(0) = 1$; $\Lambda_i(0) = b_i(0) = 0$; i = 1,…, t ; k(0) = 0; γ(0) = 0;

input: S_i , i = 0,1,…,2t-1

for r =0 **step1 until** 2t-1 **do**

begin

step 1: $\delta(r) = S_r \Lambda_0(r) + \ldots\ldots\ldots\ldots + S_{r-t} \Lambda_t(r)$

step2: $\Lambda_i(r+1) = \gamma(r) \Lambda_i(r) - \delta(r) b_{i-1}(r)$, $(i = 0, 1, \ldots, t)$

step3: if $\delta(r) \neq 0$ and $k(r) \geq 0$

then

begin

$b_i(r+1) = \Lambda_i(r)$;

$\gamma(r+1) = \delta(r)$;

$k(r+1) = -k(r)-1$;

end

else

begin

$b_i(r+1) = b_{i-1}(r)$;

$\gamma(r+1) = \gamma(r)$;

$k(r+1) = k(r)+1$;

end

end

for i=0 step1 until t-1 do

step4: $\omega_i(2t) = S_i \Lambda_0(2t) + \ldots\ldots\ldots\ldots\ldots + S_0 \Lambda_i(2t)$

output: $\Lambda_i(2t)$, i= 0,1…,t $\omega_i(2t)$, i= 0,….,(t-1)

C. Алгоритм Ченя

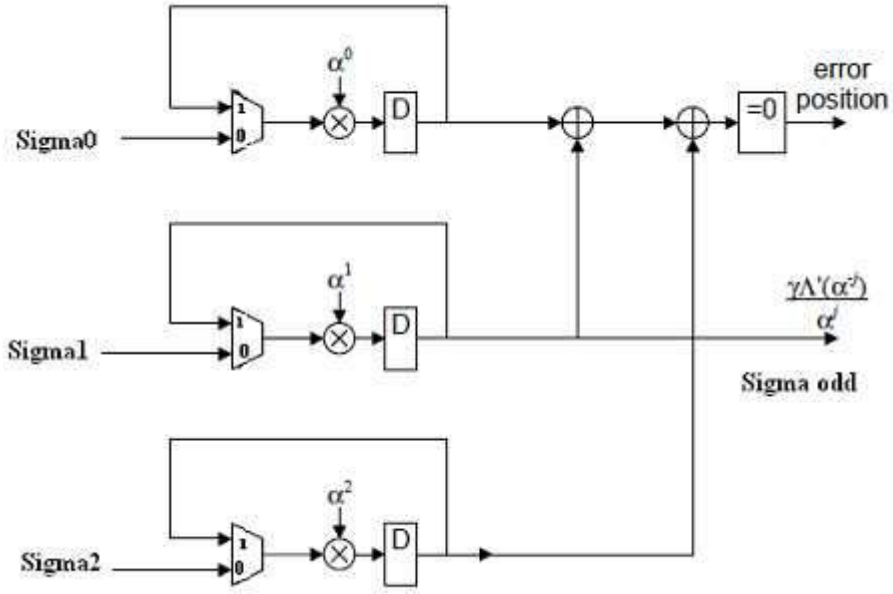


Рис. 5: Архитектура для алгоритма Ченя

Алгоритм поиска Ченя используется для оценки положения ошибки. Корни полинома локатора ошибок - это обратные места ошибки кодового слова. Этот алгоритм использует все возможные входные значения, а затем проверяет, равны ли выходы. Сумма для нечетных значений (σ1, σ 3, σ5 .....) вычисляется с одной стороны, а сумма для четных значений (σ0, σ2, σ4, σ6, ...) вычисляется в другой. Затем добавляются две суммы. Если значение суммирования равно нулю в любом такте (<n), тогда положение такта будет определять положение ошибки.

Д. Алгоритм Форни

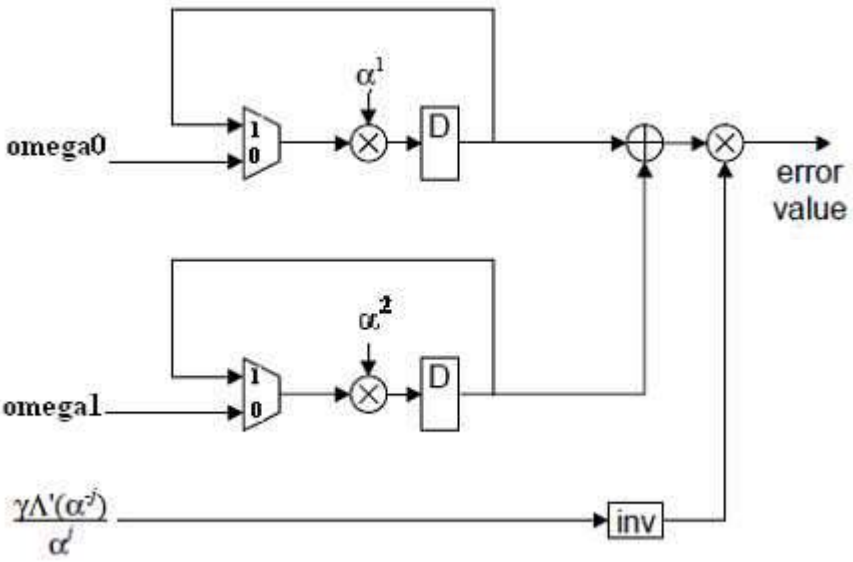


Рис. 6: Архитектура для алгоритма Форни

Алгоритм Форни используется для оценки значений ошибок. Здесь положение ошибки и коэффициенты ω(x) берутся здесь как вход. Он также использует множитель поля Галуа в качестве алгоритма поиска Ченя. Уравнение для значений ошибки определяется по формуле:

$$Y_i = \frac{\omega(x_i^{-1})}{\grave{\sigma}(x_i^{-1})} \qquad (15)$$

Здесь x_i^{-1} указывает корень, вычисленный из поиска Ченя, а σ'(x) является производной полинома локатора ошибок. Вышеприведенное уравнение можно записать в виде:

$$Y_i = \frac{\omega(x_i^{-1})(x_i^{-1})}{\sigma_{odd}(x_i^{-1})} \qquad (16)$$

$\sigma_{odd}(x_i^{-1})$ может быть найден из алгоритма поиска Ченя.

E. Коррекция ошибок

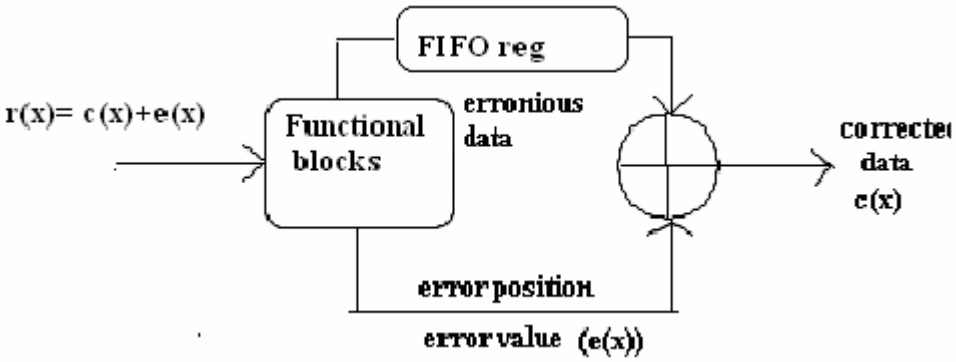


Рис. 7: Модуль коррекции ошибок

После вычисления местоположения ошибок и величин, блок коррекции ошибок принимает принятый код и выполняет операцию XOR с соответствующими значениями ошибок, вычисленными в соответствующих местах ошибок, чтобы получить исправленные символы сообщенияю

$$C(x) = R(x) + E(x) \qquad (17)$$

Поскольку символы ошибки генерируются в обратном порядке принятого кодового слова, регистр FIFO должен быть применен либо к принятому кодовому слову, либо к вектору ошибки, чтобы соответствовать

порядку байтов в обоих векторах. Выходной сигнал элемента XOR является исправленным кодовым словом декодера.

V. РЕЗУЛЬТАТ СИНТЕЗА

Каждый архитектурный модуль был реализован в Verilog и имитирован с использованием ModelSim. Проект был синтезирован инструментом Xilinx ISE 7.1i. В таблице I приведен результат синтеза RS (255,251) кодера и декодера. Целевым FPGA-устройством был Spartan3-X3S50TQ144-5.

Fun. blocks	No. of slices	No. of FFs	No. of input LUTs	No. of IOBs	Max. freq. (MHz)	Min Period (ns)
Encoder	46	40	86	67	142.55	7.02
Syndrome calculator	56	72	104	50	180.49	5.54
Key equation solver	662	64	1167	74	102.14	9.80
Chien search block	60	32	105	66	184.38	5.42
Forney block	292	24	509	90	184.38	5.42
Error corrector	28	41	41	50	183.44	5.45

VI. ВЫВОД

В документе представлен проект и реализация двухканального корректора ошибок и декодера RS (255, 251) для платформы FPGA. Предлагаемый кодек Рида-Соломона был синтезирован с использованием инструмента Xilinx ISE 7.1i. Результат показывает, что он очень эффективен, так как работает быстрее и требует меньше аппаратного обеспечения.

ЛИТЕРАТУРА

[1] S.Lin and D.J. Costello, Error Control Coding: Fundamentals and Applications. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[2] R.E. Blahut, Theory and Practice of Error Control Codes. Addison-Wesley, 1983

[3] S.B. Wicker and V.K. Bhargava, Reed Solomon Codes and Their Applications, IEEE Press, 1994.

[4] T.K. Matsushima, T. Matsushima, and S. Hirasawa, “Parallel architecture for high-speed Reed-Solomon codec,” in Proc. IEEE Int. Telecommunication. Symposium. 1998, pp. 468–473.

[5] D.V. Sarawate and N. R. Shanbhag, “High-speed architectures for Reed-Solomon decoders,” IEEE Trans. on VLSI Systems, vol. 9, no. 5, Oct. 2001, pp. 641-655.

[6] E. Mastrovito. “VLSI Architectures for Computations in Galois Fields,” Ph. D. Dissertation, Dept. of Electrical Engineering, Linkoping University, Linkoping, Sweden, 1991.

[7] L.Song, K. K. Parhi, I. Kuroda. and T. Nishitani. “Hardware/software codesign of finite field datapath for low-energy Reed-Solomon codecs,” IEEE Trans. on VLSI Systems, vol. 8. no. 2, Apr. 2000, pp. 160-172.

[8] C.Y. Lee, Y. H. Chen, C. W. Chiou and J. M. Lin, “Unified Parallel Systolic Multipliers over GF(2^m),”Journal of Computer Science and Technology, vol. 22, Jan. 2007, pp. 28-38.