УДК 621.396.41

А.Н. Кравченко

Методы проектирования параллельного BCH/CRC-кодера

Описываются наиболее практичные методы проектирования параллельного BCH/CRC-кодера. Кроме того, предложен метод проектирования последовательно-параллельных архитектур, основанных на порождающей матрице BCH-кода.

Ключевые слова: кодер, регистр сдвига, полином, порождающая матрица, метод пространства состояний.

Современные стандарты цифровой связи требуют высоких скоростей передачи данных. Для выполнения данных требований необходимо разрабатывать параллельные архитектуры, в частности и в системах обнаружения и исправления ошибок.

BCH/CRC-коды [1, 2] широко используются в современных системах связи и запоминающих устройствах как эффективные средства для обнаружения и исправления ошибок в передаваемых или считанных данных и представляют большой интерес для их эффективного и высокоскоростного аппаратного кодирования и декодирования. BCH/CRC-кодеры обычно реализуются с помощью линейного регистра сдвига с обратной связью (LFSR) [1]. Архитектура LFSR проста и может работать на высокой частоте, но в то же время имеет ограничения по производительности вследствие последовательной архитектуры. В настоящее время имеются различные методы конструирования параллельных кодеров. Среди них можно выделить методы, использующие непосредственно LFSR-регистр как базовый повторющийся элемент, инициализация которого зависит от его предыдущего состояния. Также широко используются универсальные методы, основанные на применении матриц преобразования и порождающих матриц, построенных на основе порождающего многочлена. Преимуществом последних является исключение проблемы обратной связи, что позволяет использовать конвейерную обработку в параллельном кодере, а также дает возможность параметризировать кодер. В настоящей работе рассматриваются некоторые из них, проблемы выбора конкретного метода и проблемы оптимизации не рассматриваются.

Каскадный метод построения BCH/CRC-кодера. Каскадный метод построения BCH/CRC-кодера [3] является простым методом конструирования параллельного кодера. Этот метод позволяет конструировать параллельный кодер для произвольного порождающего многочлена и произвольной ширины входных данных. Метод основан на вычислении конечного состояния LFSR-регистра, описывающего логику параллельного кодера и определяемого всеми последовательными состояниями регистра и вектором входных данных. Каждый входной информационный бит изменяет состояние регистра. Предыдущее состояние регистра и последующий входной информационный бит являются начальными условиями для последующего состояния регистра, т.е. вычисление конечного состояния строится на каскадировании состояний регистра с учетом входных информационных бит. Данная процедура может быть описана следующими уравнениями:

$$\mathbf{X}_1 = f(\mathbf{X}_0, u_{w-1}), \mathbf{X}_2 = f(\mathbf{X}_1, u_{w-2}) = f((f(\mathbf{X}_0, u_{w-1}), u_{w-2}), \ldots, \mathbf{X}_{w-1} = f[f(\mathbf{X}_0, u_{w-1}), \ldots, u_0], \quad (1)$$

где $\mathbf{X}_{i+1}$ и $\mathbf{X}_i$ являются последующими и предыдущими состояниями регистра соответственно, $\mathbf{u}$ является вектором входных данных с шириной, равной w. Пример проектирования параллельного кодера на основе данного метода будет рассмотрен ниже.

Архитектура LFSR-регистра для порождающего полинома $g(x)$ степени $m = n - k$ показана на рис. 1. Данная архитектура регистра характеризуется в литературе как тип 2 (LFSR2) [7]. Обычно в циклических кодах кодовое слово $\upsilon(x)$ является производной от двух многочленов – порождающего полинома $g(x)$ и полинома сообщения $u(x)$. Длина сообщения, входящего в кодовое слово, равна k бит. Параметр n определяет длину кодового слова. Число избыточных бит m, входящих в кодовое слово равно $m = n - k$, которое равно степени порождающего многочлена. Избыточные биты (многочлен $p(x)$) вычисляются в LFSR-регистре путем деления полинома сообщения на порождающий многочлен. При систематическом кодировании кодовое слово определяется выражением

$$\upsilon(x) = x^{n-k}u(x) + p(x) = x^{n-k}u(x) + [x^{n-k}u(x) \bmod g(x)]. \quad (2)$$

Каскадирование комбинаторной логики последовательного кодера обеспечивает простое решение параллельного кодера. Рассмотрим каскадный метод на следующем примере. Допустим, требуется спроектировать параллельный кодер с числом входных бит $w = 4$ и порождающим многочленом

$$g(x) = 1 + x^2 + x^5. \quad (3)$$

Схема последовательного кодера с порождающим многочленом (3) показана на рис. 2. Кодер включает в себя линейный регистр сдвига с обратной связью и два сумматора, вычисляющих функцию хог каждый.

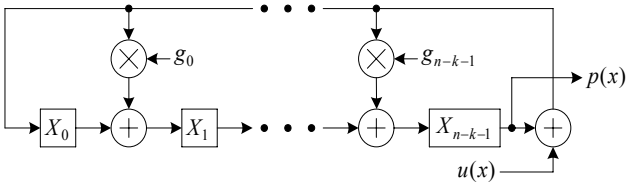

Рис. 1. Основная архитектура LFSR-регистра

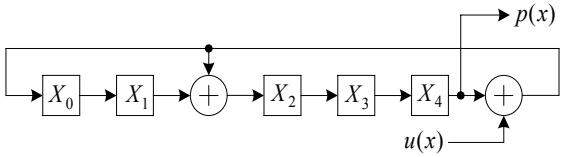

Рис. 2. Схема последовательного кодера

Параллельный кодер проектируется путем каскадирования состояний последовательного кодера:

$$\mathbf{X}_1 = f(\mathbf{X}_0, u_3), \quad \mathbf{X}_2 = f(\mathbf{X}_1, u_2) = f((f(\mathbf{X}_0, u_3), u_2), \ldots, \quad \mathbf{X}_4 = f[f(\mathbf{X}_0, u_3), \ldots, u_0].$$

1. $\mathbf{X}_1 = f(\mathbf{X}_0, u_3)$;

$$\begin{aligned} X_{0,1} &= X_{4,0} \wedge u_3; \\ X_{1,1} &= X_{0,0}; \\ X_{2,1} &= X_{1,0} \wedge X_{4,0} \wedge u_3, \\ X_{3,1} &= X_{2,0}; \\ X_{4,1} &= X_{3,0}; \end{aligned} \quad (4)$$

2. $\mathbf{X}_2 = f(\mathbf{X}_1, u_2)$;

$$\begin{aligned} X_{0,2} &= X_{4,1} \wedge u_2 = X_{3,0} \wedge u_2; \\ X_{1,2} &= X_{0,1} = X_{4,0} \wedge u_3; \\ X_{2,2} &= X_{1,1} \wedge X_{4,1} \wedge u_2 = X_{0,0} \wedge X_{3,0} \wedge u_2; \\ X_{3,2} &= X_{2,1} = X_{1,0} \wedge X_{4,0} \wedge u_3; \\ X_{4,2} &= X_{3,1} = X_{2,0}; \end{aligned} \quad (5)$$

3. $\mathbf{X}_3 = f(\mathbf{X}_2, u_1)$;

$$\begin{aligned} X_{0,3} &= X_{4,2} \wedge u_1 = X_{2,0} \wedge u_1; \\ X_{1,3} &= X_{0,2} = X_{3,0} \wedge u_2; \\ X_{2,3} &= X_{1,2} \wedge X_{4,2} \wedge u_1 = X_{4,0} \wedge u_3 \wedge X_{2,0} \wedge u_1; \\ X_{3,3} &= X_{2,2} = X_{0,0} \wedge X_{3,0} \wedge u_2; \\ X_{4,3} &= X_{3,2} = X_{1,0} \wedge X_{4,0} \wedge u_3; \end{aligned} \quad (6)$$

4. $\mathbf{X}_4 = f(\mathbf{X}_3, u_0)$;

$$\begin{aligned} X_{0,4} &= X_{4,3} \wedge u_0 = X_{1,0} \wedge X_{4,0} \wedge u_3 \wedge u_0; \\ X_{1,4} &= X_{0,3} = X_{2,0} \wedge u1; \\ X_{2,4} &= X_{1,3} \wedge X_{4,3} \wedge u_0 = X_{3,0} \wedge u_2 \wedge X_{1,0} \wedge X_{4.0} \wedge u_3 \wedge u_0; \\ X_{3,4} &= X_{2,3} = X_{4,0} \wedge u_3 \wedge X_{2,0} \wedge u_1; \\ X_{4,4} &= X_{3,3} = X_{0,0} \wedge X_{3,0} \wedge u_2. \end{aligned} \quad (7)$$

Система уравнений (7) описывает логику параллельного кодера. Знак «^», входящий в уравнения, означает логическую операцию XOR. Параллельный кодер на основе каскадного метода хорошо реализуется в высокоуровневом языке описания аппаратуры (Verilog/VHDL). Примеры «Verilog» кода для параллельного кодера можно найти в [3, 4].

Матричный метод построения BCH/CRC-кодера. Данный метод построения BCH/CRC-кодера [4] является простым методом, как и первый. Этот метод также позволяет конструировать параллельный кодер для произвольного порождающего многочлена и произвольной ширины входных данных. Данный метод детально и просто описан в [4]. На сайте Ставинова [2] http://outputlogic.com можно найти калькулятор, вычисляющий автоматически уравнения параллельного кодера. Для параллельного кодера, имеющего порождающий многочлен (3) и параметр $w=4$, калькулятор генерирует систему уравнений:

$$\begin{aligned} X_O[0] &= X_I[1] \hat{} X_I[4] \hat{} u[0] \hat{} u[3], \\ X_O[1] &= X_I[2] \hat{} u[1], \\ X_O[2] &= X_I[1] \hat{} X_I[3] \hat{} X_I[4] \hat{} u[0] \hat{} u[2] \hat{} u[3], \\ X_O[3] &= X_I[2] \hat{} X_I[4] \hat{} u[1] \hat{} u[3], \\ X_O[4] &= X_I[0] \hat{} X_I[3] \hat{} u[2], \end{aligned} \tag{8}$$

где $\mathbf{X}_I$ и $\mathbf{X}_O$ – векторы входного и выходного состояний кодера соответственно.

Как можно видеть, оба метода имеют одинаковый результат, системы уравнений (7) и (8) равны.

Метод пространстава состояний. Системный подход для проектирования параллельного кодера был предпринят в работах [5, 6]. В данных работах было введено понятие матрицы преобразования, на основе которой описывается пространство состояний последовательного кодера. Данный подход был обобщен и использован при проектировании параллельного кодера [7].

Последовательный LFSR-регистр (см. рис. 1) может рассматриваться как линейная дискретная система [7], где текущее $\mathbf{X}(i)$ и следующее состояние $\mathbf{X}(i+1)$ связаны соотношением

$$\mathbf{X}(i+1) = \mathbf{F} \cdot \mathbf{X}(i) + \mathbf{G} \cdot \mathbf{d}(i), \tag{9}$$

где $\mathbf{X} = [x_{m-1}, \ldots, x_1, x_0]$ – состояние системы; $\mathbf{d}$ – вектор входных данных. В каждом шаге, один бит сообщения $\mathbf{u}$ сдвигается в систему, и, таким образом, $d[(0) = u_{k-1}, d(1) = u_{k-2}$ и т.д. (u_{k-1} имеет самый старший разряд в сообщении). $\mathbf{F}$ – матрица размерности $m \times m$; $\mathbf{G} = [g_{m-1} \cdots g_1, g_0]^T$ – коэффициенты порождающего полинома $g(x)$. Матрица $\mathbf{F}$ описывается в общем виде выражением (10). Для порождающего полинома (3) матрица приобретает вид (11).

$$\mathbf{F} = \left[\mathbf{G} \,|\, \frac{\mathbf{I}_{m-1}}{\mathbf{0}} \right] = \begin{bmatrix} g_{m-1} & 1 & 0 & \cdots & 0 \\ g_{m-2} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1 & 0 & 0 & \cdots & 1 \\ g_0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{10}$$

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{11}$$

где $\mathbf{I}_{m-1}$ обозначает единичную матрицу размерности $(m-1) \times (m-1)$. Для решения уравнения (9) используется рекурсивный метод, который выполняется следующим образом:

$$\begin{aligned} \mathbf{X}(1) &= \mathbf{F} \cdot \mathbf{X}(0) + \mathbf{G} \cdot d(0) = \mathbf{F} \cdot (\mathbf{X}(0) + [0,0,\ldots,d(0)]^T); \\ \mathbf{X}(2) &= \mathbf{F} \cdot \mathbf{X}(1) + \mathbf{G} \cdot d(1) = \mathbf{F}^2 (\mathbf{X}(0) + [0,0,\ldots,d(1),(d(0)]^T); \\ &\vdots \\ \mathbf{X}(w) &= \mathbf{F}^w \cdot (\mathbf{X}(0) + [0,\ldots,0 \,|\, d(w-1),\ldots,d(0)]^T). \end{aligned} \tag{12}$$

Уравнение (12) справедливо для случая, когда $w \le m$. В вектор входных данных введены уравнивающие нули, для получения порядка вектора, равного m. Введем обозначение $\mathbf{D} = [0,\ldots,0 \,|\, d(w-1),\ldots,d(0)]^T$, тогда уравнение (12) приобретает вид $\mathbf{X}(w) = \mathbf{F}^w \cdot (\mathbf{X}(0) + \mathbf{D}(0))$. Данное уравнение используется для вычисления состояния параллельного кодера для первой группы w бит сообщения $\mathbf{u}$. Для следующей группы используется уравнение $\mathbf{X}(2w) = \mathbf{F}^w \cdot (\mathbf{X}(w) + \mathbf{D}(1))$. Рекурсивную процедуру вычисления вектора $p(x)$ (избыточных бит) описывает уравнение

$$\mathbf{X}(qw) = \mathbf{F}^w \cdot (\mathbf{X}((q-1)w) + \mathbf{D}(q)), \quad (13)$$

где $q = 1,2,\ldots,l$. Параметр l определяет число групп в сообщении $\mathbf{u}$, $l = k/w$. Простой способ вычисления матрицы $\mathbf{F}^m$ (матрицы преобразования) показывает следующий рекурсивный алгоритм:

$$\mathbf{F}^m = \mathbf{F} \cdot \mathbf{F};\ \text{for}\ i = 1{:}\,m-2\ \ \mathbf{F}^m = \mathbf{F}^m \cdot \mathbf{F};\ \text{end}.$$

Матрица $\mathbf{F}^w$, входящая в уравнение (12), может быть легко получена из матрицы $\mathbf{F}^m$ следующим образом. Первые w столбцов матрицы $\mathbf{F}^w$ есть последние столбцы матрицы $\mathbf{F}^m$. Верхняя часть матрицы $\mathbf{F}^w$ – единичная матрица $\mathbf{I}_{m-w}$ и нижняя часть заполняются нулями. Примеры вычисленных матриц для порождающего полинома (3) представлены выражениями (14) и (15) соответственно. Пример аппаратной реализации данного метода можно найти в [8].

$$\mathbf{F}^5 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad (14)$$

$$\mathbf{F}^4 = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (15)$$

Вычисленные значения избыточных бит **R** для входного сообщения **u** = FED03F, для порождающего многочлена (3) и $w = 4$ равны **R** = ‘13 hex. Одинаковый результат (в обращенной форме) дает также кодер, спроектированный на основе уравнений (8). В параллельном кодере, спроектированном на базе LFSR-регистра типа 1, избыточные биты вычисляются в соответствии со следующим рекурсивным выражением [7]:

$$\mathbf{X}(qw) = \mathbf{F}^w \cdot \mathbf{X}((q-1)w) + \mathbf{D}(q),\ q = 1,2,\ldots,l. \quad (16)$$

Данное выражение получается из (9), если положить $\mathbf{G} = [0\,0\cdots 1]^T$.

Метод проектирования, основанный на порождающей матрице. Как и в предыдущем методе, исходное сообщение $u(x)$ представляется в виде l групп, $l = k/w$:

$$u(x) = u_0 + u_1 x + \cdots + u_{k-1}x^{k-1} = d_0(x) + d_1(x)x^w + d_2(x)x^{2w} + \ldots + d_{l-1}(x)x^{w(l-1)}, \quad (17)$$

где

$$d_q(x) = d_{qw} + d_{qw+1}x + d_{qw+2}x^2 + \ldots + d_{(q+1)w-1}x^{w-1}. \quad (18)$$

С учетом выражения (18) многочлен $p(x)$, входящий в уравнение (2), может быть записан как

$$p(x) = x^{n-k}u(x) \bmod g(x) = x^m(d_0(x) + d_1(x)x^w + d_2(x)x^{2w} + \ldots + d_{l-1}(x)x^{w(l-1)}) \bmod g(x) =$$

$$= x^m d_0(x) \bmod g(x) + x^m d_1(x)x^w \bmod g(x) + x^m d_2(x)x^{2w} \bmod g(x) + \ldots + x^m d_{l-1}(x)x^{w(l-1)}) \bmod g(x). \quad (19)$$

Рассмотрим детально первый член $x^m d_0(x) \bmod g(x)$, входящий в многочлен $p(x)$ (19):

$$x^m d_0(x) \bmod g(x) = x^m(d_0 + d_1 x + d_2 x^2 + \ldots + d_{w-1}x^{w-1}) \bmod g(x) =$$

$$= x^m \bmod g(x) d_0 + x^{m+1} \bmod g(x) d_1 + x^{m+2} \bmod g(x) d_2 + \ldots + x^{m+w} \bmod g(x) d_{w-1}, \quad (20)$$

где $\mathbf{g}_0 = x^m \bmod g(x)$, $\mathbf{g}_1 = x^{m+1} \bmod g(x)$, $\mathbf{g}_2 = x^{m+2} \bmod g(x)$, …, $\mathbf{g}_{w-1} = x^{m+w} \bmod g(x)$ являются векторами коэффициентов порождающей матрицы $\mathbf{G} = [\mathbf{g}_0\,\mathbf{g}_1 \ldots \mathbf{g}_{w-1}]^T$, вычисленной на основе порождающего многочлена $g(x)$. С учетом коэффициентов порождающей матрицы выражение (19) можно записать в виде

$$p(x) = (\mathbf{g}_0 d_0 + \mathbf{g}_1 d_1 + \ldots \mathbf{g}_{w-1} d_{w-1}) + (\mathbf{g}_0 d_w + \mathbf{g}_1 d_{w+1} + \ldots \mathbf{g}_{w-1} d_{2w-1})x + (\mathbf{g}_0 d_{2w} + \mathbf{g}_1 d_{2w+1} +$$

$$+ \ldots \mathbf{g}_{w-1} d_{3w-1})x^2 + \ldots + (\mathbf{g}_0 d_{qw} + \mathbf{g}_1 d_{qw+1} + \ldots \mathbf{g}_{w-1} d_{(q+1)w-1})x^{w(l-1)}. \quad (21)$$

Вычисление $p(x)$ можно выполнить рекурсивным способом. Допустим, что параметр w является кратным m. Тогда вектор $\mathbf{p}$ можно представить в виде групп $\mathbf{p} = [\mathbf{P}^0\,\mathbf{P}^1 \ldots \mathbf{P}^{m/w-1}]$. Введем следующие обозначения сумм:

$S_j^0 = \sum_{r=0}^{w-1} X_r g_{j,r}$, $S_j^1 = \sum_{r=0}^{w-1} X_r g_{(w+j),r}$, $S_j^{m/w-1} = \sum_{r=0}^{w-1} X_r g_{[(m/w-1)w]+j,r}$, for $j = 0,1,\ldots w-1$, (22)

где $\mathbf{X} = \mathbf{d}_q + \mathbf{P}^{m/w-1}$, для $q = 0,1, \ldots l-1$.

С учетом введенных обозначений запишем следующий алгоритм вычисления **p**:

$\mathbf{p} = [\mathbf{P}^0\ \mathbf{P}^1 \ldots \mathbf{P}^{m/w-1}] = 0$

$\mathbf{s} = [\mathbf{S}^0\ \mathbf{S}^1 \ldots \mathbf{S}^{m/w-1}] = 0$

for $q = 0,1,\ldots l-1$

$\mathbf{X} = \mathbf{d}_q + \mathbf{P}^{m/w-1}$

$\mathbf{P}^0 = \mathbf{S}^0$

for $z = 1,\ldots,m/w-1$

$\mathbf{P}^z = \mathbf{P}^{z-1} + \mathbf{S}^z$

end

end

Данный метод легко реализуется в аппаратуре. Рассмотрим конкретный пример. Допустим, необходимо спроектировать последовательно-параллельный кодер для BCH кода (128, 120, 8), имеющего порождающий многочлен $g(x) = x^0 + x^4 + x^5 + x^6 + x^8$. В данном примере $m = 8,\ w = 4$.

Вычисленные коэффициенты порождающей матрицы имеют значения:

$$\mathbf{g}_0 = x^8 \bmod g(x) = [1\ 0\ 0\ 0\ 1\ 1\ 1\ 0]; \quad \mathbf{g}_1 = x^9 \bmod g(x) = [0\ 1\ 0\ 0\ 0\ 1\ 1\ 1];$$

$$\mathbf{g}_2 = x^{10} \bmod g(x) = [1\ 0\ 1\ 0\ 1\ 1\ 0\ 1]; \quad \mathbf{g}_3 = x^{11} \bmod g(x) = [1\ 1\ 0\ 1\ 1\ 0\ 0\ 0].$$

Значения избыточных бит, вычисленных для сообщения:

u = 5584BA72570A961220150291BD3AF8 равны **P** =’48’, hex.

Полученное кодовое слово в систематической форме имеет вид

υ = 5584BA72570A961220150291BD3AF848.

Синдромы [1], вычисленные для данного кодового слова, в отсутствие ошибок, равны нулю: $S_0 = 0,\ S_1 = 0$.

Архитектура кодера, реализующая данный алгоритм, изображена на рис. 3.

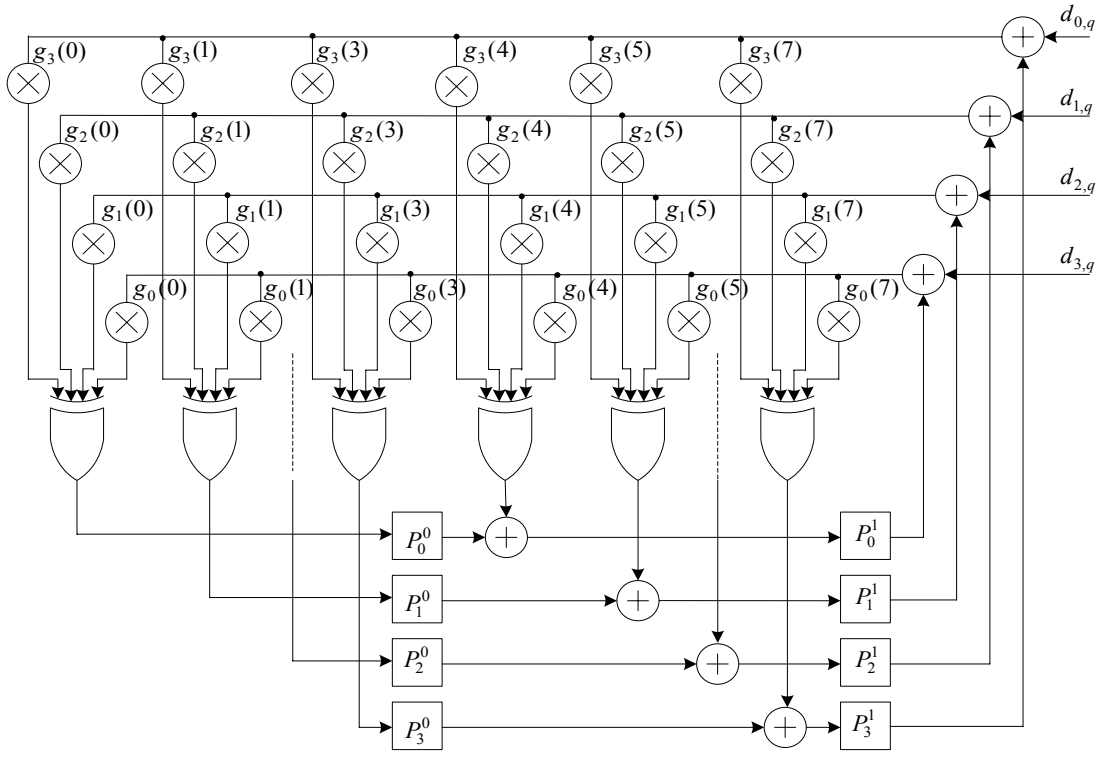

Рис. 3. Архитектура кодера для BCH (128, 120, 8) кода

В настоящее время существуют различные вариации вышеперечисленных методов, нацеленных на реализацию в конкретной аппаратуре. Например, в работе [9] рассматривается возможность параметризации параллельного кодера, в работе [10] – возможность применения конвейерной обработки. В работе [11] затронуты вопросы оптимизации.

Заключение. В работе представлены методы проектирования параллельного BCH/CRC-кодера, позволяющие спроектировать кодер соответствующей конфигурации на основе произвольной ширины входных данных и заданного порождающего многочлена. Показан также метод проектирования, использующий вычисленные коэффициенты порождающей матрицы, как начальные значения в параллельном матричном процессоре, выполняющем операции умножения и сложения в поле Галуа. Кодер, спроектированный на основе данного метода, имеет регулярную структуру и легко реализируется в аппаратуре, которая дает возможность параметризации и оптимизации параллельного кодера.

Литература

1. Блейхут Р. Теория и практика кодов, контролирующих ошибки / пер. с англ., под ред. К.Ш. Зигангирова. – М.: Мир, 1986. – 289 с.

2. Вернер М. Основы кодирования / пер. с нем., Д.К. Зигангирова. – М.: Техносфера, 2004. – 284 с.

3. Sprachmann M. Automatic Generation of Parallel CRC Circuits // IEEE Design & Test of Computers. – 2001. – May–June. – P. 108–114.

4. Stavinov E. A practical Parallel CRC Generation Method, CIRCUIT CELLAR, – 2010. – January – Issue 234. P. 38–45 [Электронный ресурс]. – Режим доступа: http://outputlogic.com/my-stuff/circuitcellar-january-2010-crc.pdf.

5. Pei T.B. High-Speed Parallel CRC in VLSI / T.B. Pei, C. Zukowski // IEEE Trans. on Communications. – 1992. –Vol. 40, No 4. – P. 653–657.

6. Shien M.D. A Systematic Approach for Parallel CRC Computations / M.D. Shien, M.H. Sheu, H.F. Lo // Journal of information science and engineering. – 2001. – No. 17. –P. 445–461.

7. Campobello G. Parallel CRC Realization / G. Campobello, G. Patane, M. Russo // IEEE Trans. on Computers. – 2003. – Vol. 52, No. 10. – P. 1312–1319.

8. Qu X. A Novel Least Significant Bit First Processing Parallel CRC Circuit / X. Qu, Z. Cao, Z. Yang // Hindawi Publishing Corporation, Advances in Mechanical Engineering. – 2013. – Article ID 859317, 5 p. [Электронный ресурс]. – Режим доступа: http://www.hindawi.com/journals/ame/2013/859317/abs/

9. Grymel M. A Novel Programmable Parallel CRC Circuit / M. Grymel, S.B. Furber // IEEE Trans. on VLSI. – 2011. – Vol. 19, No. 10. – P. 1898–1902.

10. Ayinava M. High Throughput VLSI Architectures for CRC/BCH Encoders and FFT computations // A thesis submitted to the faculty of the graduate school of the university of Minnesota. – December 2010 [Электронный ресурс]. – Режим доступа: http://conservancy.umn.edu/bitstream/103800/1/Ayinala_Manohar_January2011.pdf

11. Optimized design for high-speed parallel BCH encoder / Z. Jun, W. Zhu-Gong, H. Qing-Shene, X. Jie // VLSI Design and Video Technology 2005. – Proceedings of 2005 IEEE International Workshop on. – 2005. – May 28–30. – P. 97–100.

Кравченко Александр Николаевич
Канд. техн. наук, системный архитектор Ubiso GmbH,
Hermann-Schwer-Str. 3 78048 VS-Villingen, Germany
Тел.: +49(0) 7-721-944-76-77
Эл. почта: alexander.kravtchenko@ubiso.com

Kravtchenko A.N.
Methods for design of parallel BCH/CRC-encoder

The practical methods of parallel BCH/CRC-encoder design are described. Further the method of design for serial-parallel architectures is proposed. The method is based on the generator matrix of BCH code.
Keywords: encoder, shift register, generator matrix, state space representation.