УДК 621.391.251:681.3.06

**Кубицкий В.И.**

# СЛОЖНОСТЬ АЛГОРИТМОВ КОДИРОВАНИЯ КОДОВ ЛАГРАНЖА

**ГосНИИ «Аэронавигация» (Россия, Москва)**

*Определена сложность (количество операций в конечных полях) алгоритмов кодирования полных и неполных кодов Лагранжа. Проведён сравнительный анализ различных алгоритмов кодирования этих кодов.*

## Введение

В [1] предложено кодирование, основанное на интерполяционной формуле Лагранжа. Код, полученный в результате такого кодирования, назван кодом Лагранжа. В [2] процедуры кодирования были модифицированы и получили названия параллельного, последовательного и параллельно-последовательного алгоритмов кодирования. В [3] код, получаемый с использованием этих алгоритмов, назван полным кодом Лагранжа и разработаны неполные коды Лагранжа. В [4-6] показано, что при изменении значений одного или нескольких информационных символов необходимо производить перекодирование кодового слова. В [5] разработаны параллельный, последовательный и параллельно-последовательный алгоритмы перекодирования. Для выбора алгоритмов кодирования/перекодирования, лучших при определённых условиях применения, необходимо провести их сравнительный анализ.

В данной статье определим количество модульных операций сложения $N_{\oplus}$, умножения $N_{\otimes}$ и инвертирования $N_{\textcircled{\ulcorner}}$, которое необходимо выполнить в конечном поле $GF(2^m)$ при кодировании/перекодировании, и проведём их сравнение для различных алгоритмов. Количество модульных операций назовём сложностью алгоритмов кодирования/перекодирования. Обозначения в статье соответствуют обозначениям, принятым в [2], [3] и [5].

## 1. Сложность алгоритмов кодирования

*1.1. Сложность алгоритмов кодирования полных кодов*

Для *параллельного алгоритма кодирования* сложность равна:

$$N_{\oplus}=\begin{cases}2r\,(n-1)-r^2\ , & \text{при } r>1\,,\\ n-2\,, & \text{при } r=1\,,\end{cases}$$

$$N_{\otimes}=r\,(r-1)(n-r+1)\ ,$$

$$N_{\textcircled{\ulcorner}}=r\ .$$

Здесь $n=k+r$ – длина кодового слова, $k$ – количество информационных символов кодового слова, $r$ – количество контрольных символов кодового слова.

Расчёт количества операций произведён для случая, когда необходимо вычислять коэффициенты Лагранжа $L^{(i)}(x)$ (т.е. при $L^{(i)}(x)\neq const$), и с учётом наличия $r$ регистров памяти для хранения величин $a_{il}=x_i-\beta_l$ ($i=\overline{0,s}$, $l=\overline{1,r}$). С учётом наличия памяти для хранения $n$ узлов интерполирования необходимо иметь ($n+r$) регистров.

Количество операций в поле $GF(2^m)$ для *параллельного алгоритма кодирования при* $L^{(i)}(x)=const$ следующее:

$$N_{\oplus}=(n-r-1)\,r\,,$$

$$N_{\otimes}=\begin{cases}(n-r)\,r\,, & \text{при } r>1\,,\\ 0\,, & \text{при } r=1\,.\end{cases}$$

Для хранения коэффициентов $L^{(i)}(x)$ требуется $r\,(n-r)$ регистров.

Сложность *последовательного алгоритма кодирования при* $L^{(i)}(x)\neq const$ составляет:

$$N_{\oplus} = n(2r-1) - r(r+1),$$

$$N_{\otimes} = (n-1)(r-1),$$

$$N_{\copyright} = r-1.$$

Расчёт количества операций производился с учётом наличия $(r-1)$ регистров памяти для хранения величин $b_{jl} = \beta_j - \beta_l$ ( $j = \overline{1, r-1}$, $l = \overline{r, j+2}$ ). С учётом наличия памяти для хранения $n$ узлов интерполирования необходимо иметь $(n+r-1)$ регистров.

Количество операций в поле $GF(2^m)$ для *последовательного алгоритма кодирования при* $L^{(i)}(x) = const$ равно:

$$N_{\oplus} = (2n-r-3)\, r/2 ,$$

$$N_{\otimes} = (2n-r-2)(r-1)/2 .$$

Для хранения коэффициентов $L^{(i)}(x)$ нужно $(r-1)(2n-r-2)/2$ регистров.

Для *параллельно-последовательного алгоритма кодирования при* $L^{(i)}(x) \neq const$ количество операций в поле $GF(2^m)$ будет следующим:

$$N_{\oplus} = 2r(n-1) - r^2,$$

$$N_{\otimes} = (r-1)^2(n-r+1),$$

$$N_{\copyright} = r-1.$$

Расчёт количества операций производился с учётом наличия $r$ регистров памяти для хранения величин $a_{il} = x_i - \beta_l$ ($i = \overline{0, s}$, $l = \overline{1, r}$ ). С учётом наличия памяти для хранения $n$ узлов интерполирования необходимо иметь $(n + r)$ регистров.

Количество операций в поле $GF(2^m)$ для *параллельно-последовательного алгоритма кодирования при* $L^{(i)}(x) = const$ равно:

$$N_{\oplus} = r(n-r) - 1,$$

$$N_{\otimes} = (r-1)(n-r).$$

Для хранения коэффициентов $L^{(i)}(x)$ необходимо иметь $(r-1)(n-r)$ регистров.

*1.2. Сложность алгоритмов перекодирования*

Алгоритмы перекодирования предполагают наличие $z$ регистров памяти для хранения $z$ изменяемых узлов интерполирования, а также $r$ регистров для хранения величин $a_{\xi l} = \tilde{x}_{\xi} - \beta_l$ ( $\xi = \overline{1, z}$, $l = \overline{1, r}$ ) (для параллельного и параллельно-последовательного алгоритмов) и $(r-1)$ регистров для хранения величин $b_{jl} = \beta_j - \beta_l$ ( $j = \overline{1, r-1}$, $l = \overline{r, j+2}$ ) (для последовательного алгоритма).

При изменении значений только информационных символов номера контрольных узлов остаются прежними и поэтому можно принять $\gamma_j = const$. При этом для хранения величин $\gamma_j$ требуется $r$ регистров (для параллельного алгоритма: $\gamma_j = -1/\prod_{l=1, l\neq j}^{r}(\beta_j - \beta_l)$, $j = \overline{1, r}$ ) и $(r-1)$ регистров (для последовательного: $\gamma_j = -1/\prod_{l=j+1}^{r}(\beta_j - \beta_l)$, $j = \overline{1, r-1}$ и параллельно-последовательного: $\gamma_j = -1/\prod_{l=1, l\neq j}^{r}(\beta_j - \beta_l)$, $j = \overline{1, r-1}$ алгоритмов), $r(r-1)/2$ регистров – для хранения величин $b_{jl}$ (для последовательного алгоритма), $r$ регистров – для хранения величин $a_{\xi l}$ (для параллельного и параллельно-последовательного алгоритмов), $z$ регистров – для хранения $z$ изменяемых узлов интерполирования.

Количество операций в конечном поле $GF(2^m)$, которое необходимо выполнить для перекодирования, составляет:

1) для *параллельного алгоритма*:

а) при $\gamma_j \neq const$:

$$N_{\oplus} = z(2r+1) + r(r-1),$$

$$N_{\otimes} = r(r-1)(z+1),$$

$$N_{\copyright} = r;$$

б) при $\gamma_j = const$:

$$N_{\oplus} = z(2r+1),$$

$N_{\otimes} = r[z(r-1)+1]$;

2) для *последовательного алгоритма*:

а) при $\gamma_j \neq const$:

$N_{\oplus} = 2zr + (r^2 - 1)$,

$N_{\otimes} = (r-1)(z+r-1)$,

$N_{\circledcirc} = r-1$;

б) при $\gamma_j = const$:

$N_{\oplus} = 2zr + (r-1)(r+2)/2$,

$N_{\otimes} = (r-1)(2z+r)/2$;

3) для *параллельно-последовательного алгоритма*:

а) при $\gamma_j \neq const$:

$N_{\oplus} = z(2r+1) + r(r-1)$,

$N_{\oplus} = (r-1)(r-1)(z+1)$,

$N_{\circledcirc} = r-1$;

б) при $\gamma_j = const$:

$N_{\oplus} = z(2r+1) + (r-1)$,

$N_{\otimes} = (r-1)(z+2)$.

*1.3. Сложность алгоритмов кодирования неполных кодов*

Количество операций, которое необходимо выполнить в поле $GF(2^m)$ для кодирования неполного кода *параллельным алгоритмом при* $L^{(i)}(x) \neq const$, равно:

$N_{\oplus} = (k+r)(2r+e) - r(r+2)$,

$N_{\otimes} = r(r-1)(k+1) + (k+r)\,e$,

$N_{\circledcirc} = r$.

При этом для хранения величин $\alpha_i = f_i \prod_{v_\xi \in V} (x_i - v_\xi)$ $(i = \overline{0, s}, \xi = \overline{1, e})$ и $a_{il} = x_i - \beta_l$ $(i = \overline{0, s}, \ l = \overline{1, r})$ требуется соответственно один и $r$ регистров.

При $L^{(i)}(x) = const$ количество операций в поле $GF(2^m)$ такое же, как для параллельного алгоритма полного кода:

$N_{\oplus} = (n-r-1)\,r$,

$$N_{\otimes} = \begin{cases} (n-r)\,r, & \text{при } r > 1, \\ 0, & \text{при } r = 1, \end{cases}$$

где $n = k + r + e$.

Для хранения коэффициентов $L^{(i)}(x)$ необходимо иметь $kr$ регистров памяти.

Количество операций, которое необходимо выполнить в поле $GF(2^m)$ для *последовательного алгоритма кодирования* неполного кода *при* $L^{(i)}(x) \neq const$, будет следующим:

$N_{\oplus} = (k+r)\,(2r+e-1) - r(r+1)$,

$N_{\otimes} = (k+r)(r+e-1)$,

$N_{\circledcirc} = r$.

Расчёт производился с учётом наличия $(r-1)$ регистров для хранения величин $b_{jl} = \beta_j - \beta_l$ $(j = \overline{1, r-1}, \ l = \overline{r, j+2})$, 1 регистр – для $\nabla_j = \prod_{v_\xi \in V} (\beta_j - v_\xi)$ $(j = \overline{1, r}$, $\xi = \overline{1, e})$ и 1 регистр – для $\alpha_i = f_i \prod_{v_\xi \in V} (x_i - v_\xi)$ $(i = \overline{0, s}, \xi = \overline{1, e})$.

При $L^{(i)}(x) = const$ потребуется выполнить модульных операций сложения такое же количество, как для последовательного алгоритма полного кода:

$$N_{\oplus} = (2n - r - 3)\,r/2,$$

где $n = k + r + e$,

а количество операций умножения составляет:

$$N_{\otimes} = (2k + r - 2)(r-1)/2 + (k + r - 1).$$

Для хранения коэффициентов $L^{(i)}(x)$ необходимо иметь $r(2k+r-1)/2$ регистров памяти.

Количество операций в поле $GF(2^m)$ для кодирования неполного кода *параллельно-последовательным алгоритмом при* $L^{(i)}(x) \neq const$ равно:

$N_{\oplus} = (k+r)(2r+e) - r(r+1) - 1$,

$N_{\otimes} = (r-1)[r(k+1)-1] + (k+r)\,e$,

$N_{\circledcirc} = r$.

Для хранения величин $\alpha_i$ и $\nabla_j$ необходимо иметь по одному регистру памяти и $r$ регистров – для хранения величин $a_{il}$.

При $L^{(i)}(x) = const$ потребуется выполнить модульных операций сложения такое же количество, как для параллельно-последовательного алгоритма кодирования полного кода:

$$N_{\oplus} = r(n-r) - 1,$$

где $n = k + r + e$,
а количество операций умножения составляет:

$$N_{\otimes} = r(k+1) - 1.$$

Для хранения коэффициентов $L^{(i)}(x)$ необходимо иметь $r(k+1)-1$ регистров памяти.

## *2. Сравнение сложностей алгоритмов*

*2.1. Сравнение сложностей алгоритмов кодирования полного кода*

Сравним параллельный и последовательный алгоритмы кодирования. Получим:

1) при $L^{(i)}(x) \neq const$:

а) количество операций в конечном поле для последовательного алгоритма меньше на величину $(n-r)$ - для сложения, на величину $(r-1)^2(n-r)+(r-1)$ - для умножения, на 1 – для инвертирования;

2) при $L^{(i)}(x) = const$:

а) количество операций сложения для параллельного алгоритма меньше на величину $r(r-1)/2$, количество операций умножения меньше для последовательного алгоритма на величину $(n-1)-r(r-1)/2$ при $r < (1+\sqrt{8n-7})/2$;

б) количество регистров для хранения коэффициентов $L^{(i)}(x)$ для последовательного алгоритма меньше на величину $(n-1)-r(r-1)/2$ при $r < (1+\sqrt{8n-7})/2$.

В результате сравнения параллельного и параллельно-последовательного алгоритмов кодирования имеем:

1) при $L^{(i)}(x) \neq const$:

а) для параллельно-последовательного алгоритма выполняется меньше операций умножения на величину $(r-1)(n-r+1)$ и на 1 операцию инвертирования, сложений требуется одинаковое количество;

2) при $L^{(i)}(x) = const$:

а) для параллельно-последовательного алгоритма требуется операций умножения меньше на величину $(n-r)$, но больше операций сложения на величину $(r-1)$;

б) количество регистров для хранения коэффициентов $L^{(i)}(x)$ для параллельно-последовательного алгоритма меньше на величину $(n-r)$.

Сравнение последовательного и параллельно-последовательного алгоритмов кодирования даёт следующий результат:

1) при $L^{(i)}(x) \neq const$:

а) для последовательного алгоритма требуется меньше операций сложения на величину $(n-r)$ и операций умножения на величину $(r-1)(r-2)(n-r)$, инвертирований выполняется одинаковое количество;

2) при $L^{(i)}(x) = const$:

а) для параллельно-последовательного алгоритма выполняется меньше на $r(r-3)/2+1$ операций сложения и на $(r-1)(r-2)/2$ операций умножения;

б) количество регистров для хранения коэффициентов $L^{(i)}(x)$ меньше для параллельно-последовательного алгоритма на величину $(r-1)(r-2)/2$.

Сравним теперь алгоритмы кодирования кодов Лагранжа с процедурой кодирования кодов Рида-Соломона.

При систематическом кодировании РС-кодов необходимо выполнить $r(n-r)$ модульных умножений и столько же сложений [7]. Это на одну операцию сложения и на $(n-r)$ операций умножения больше, чем для параллельно-последовательного алгоритма кодирования кодов Лагранжа при $L^{(i)}(x) = const$.

Для параллельного алгоритма кодирования количество сложений на величину $r$ меньше, чем для РС-кодов, количество умножений – одинаковое. Очевидно, что по количеству операций в конечном поле предпочтение отдаётся параллельно-последовательному алгоритму кодирования кодов Лагранжа при $L^{(i)}(x) = const$.

Применение последовательного алгоритма кодирования кодов Лагранжа при $L^{(i)}(x) \neq const$ требует выполнить для всех $r$ на $n(r-1)-r$ операций сложения больше, чем при кодировании РС-кодов, а операций умножения – на $(n-1)-r(r-1)$ меньше, но для $r < (1+\sqrt{4n-3})/2$.

Если кодирование РС-кодов выполняется с помощью универсального компьютера, то, применяя быстрое преобразование Матсона-Соломона, можно добиться меньшего количества операций [8]. Но при этом быстрая процедура систематического кодирования требует специального размещения проверочных позиций, при котором многочлен локаторов проверочных символов имеет ограниченный вес. К тому же не для всех значений $r$ количество операций будет меньшим.

Оценка сложности для быстрой процедуры систематического кодирования РС-кодов следующая: $2cn\ln r + r$, где $r$ - делитель $n$, $c$ - константа ($c = 1/\ln 2 = 1{,}443$).

Класс систематических РС-кодов, для которых достигается эта оценка, ограничена набором делителей $n$. Сложность кодирования с применением параллельно-последовательного алгоритма будет меньше приведенной сложности РС-кодов при выполнении условий:

$k/n < (2c\ln r + 1)/r$ – для умножения,

$k/n \lesssim (2c\ln r + 1)/(r+1)$ – для сложения.

Асимптотически самой быстрой процедурой кодирования РС-кодов является процедура несистематического кодирования, состоящая в вычислении преобразования Матсона-Соломона для последовательности информационных символов. Для этой процедуры требуется операций: $N \le cn\ln k$. Сложность кодирования кода Лагранжа с применением параллельно-последовательного алгоритма будет меньше этой величины, если удовлетворяются условия:

$k/n\ln k < c/(r-1)$ – для умножения,

$k/(cn\ln k + 1) < 1/r$ – для сложения.

*2.2. Сравнение сложностей алгоритмов перекодирования*

Сравним параллельный и последовательный алгоритмы перекодирования.

Для $\gamma_j \neq const$:

1) $\Delta_{\oplus} = z-(r-1)$. Здесь $\Delta_{\oplus} \ge 0$ для $z \ge r-1$.

2) $\Delta_{\otimes} = (r-1)[z(r-1)+1]$. Здесь $\Delta_{\oplus} > 0$ для всех $z > 0$.

3) $\Delta_{\Gamma} = 1$.

По количеству операций в конечном поле предпочтение при $\gamma_j \neq const$ отдаётся последовательному алгоритму перекодирования.

Для $\gamma_j = const$:

1) $\Delta_{\oplus} = z-(r-1)(r+2)/2$. Здесь $\Delta_{\oplus} \ge 0$ для $z \ge (r-1)(r+2)/2$.

2) $\Delta_{\otimes} = (r-1)[z(r-1)-r/2]+r$. Здесь $\Delta_{\otimes} \ge 0$ для $z \ge 1$.

По количеству операций умножения в конечном поле предпочтение при $\gamma_j = const$ отдаётся последовательному алгоритму перекодирования.

Сравним последовательный и параллельно-последовательный алгоритмы перекодирования.

Для $\gamma_j \neq const$:

1) $\Delta_{\oplus} = z-(r-1)$. Здесь $\Delta_{\oplus} \ge 0$ для $z \ge r-1$.

2) $\Delta_{\otimes} = (r-1)(r-2)z$. Здесь $\Delta_{\oplus} > 0$ для всех $z > 0$ при $r > 2$.

3) $\Delta_{\Gamma} = 0$.

По количеству операций в конечном поле предпочтение при $\gamma_j \neq const$ отдаётся последовательному алгоритму перекодирования.

Для $\gamma_j = const$:

1) $\Delta_{\oplus} = z-(r-1)r/2$. Здесь $\Delta_{\oplus} \geq 0$ для $z \geq r(r-1)$.

2) $\Delta_{\otimes} = (r-1)(r-2)(2z-1)/2$. Здесь $\Delta_{\oplus} > 0$ для всех $z > 0$ при $r > 2$.

Сравним параллельный и параллельно-последовательный алгоритмы перекодирования.

Для $\gamma_j \neq const$:

1) $\Delta_{\oplus} = 0$.

2) $\Delta_{\otimes} = (r-1)(r-2)z$. Здесь $\Delta_{\oplus} > 0$ для всех $z > 0$ при $r > 2$.

3) $\Delta_{\copyright} = 1$.

По количеству операций в конечном поле предпочтение при $\gamma_j \neq const$ отдаётся параллельно-последовательному алгоритму перекодирования.

Для $\gamma_j = const$:

1) $\Delta_{\oplus} = -(r-1)$. Здесь всегда $\Delta_{\oplus} \leq 0$.

2) $\Delta_{\otimes} = z(r-1)+1$. Здесь $\Delta_{\oplus} > 0$ для всех $z > 0$ при $r > 1$.

При $\gamma_j \neq const$ по количеству операций сложения в конечном поле предпочтение отдаётся параллельному алгоритму, по количеству операций умножения – параллельно-последовательному алгоритму перекодирования.

Определим путём сравнения, имеется ли выигрыш для предложенных алгоритмов перекодирования по сравнению с тем, если бы пришлось производить перекодирование с учётом всех информационных символов в соответствии с алгоритмами кодирования.

1) Для параллельных алгоритмов.

а) При $\gamma_j \neq const$:

$\Delta_{\oplus} = r[2(n-r)-1]-(2r+1)z]$,

$\Delta_{\otimes} = r(r-1)(n-r-z)$, $\Delta_{\copyright} = 0$,

где $n=k+r$ - длина кодового слова.

Здесь $\Delta_{\oplus} > 0$ для $z < r[2(n-r)-1]/(2r+1)$, $\Delta_{\otimes} > 0$ для $z < n-r$.

б) При $\gamma_j = const$:

$\Delta_{\oplus} = r(n-r-1)-z(2r+1)$,

$\Delta_{\otimes} = r[(n-r-1)-z(r-1)$.

Здесь $\Delta_{\oplus} > 0$ для $z < (n-r-1)/(2r+1)$, $\Delta_{\otimes} > 0$ для $z < (n-r-1)/(r-1)$.

2) Для последовательных алгоритмов.

а) При $\gamma_j \neq const$:

$\Delta_{\oplus} = n(2r-1)+(r+1)-2zr$,

$\Delta_{\otimes} = (r-1)(n-r-z)$, $\Delta_{\copyright} = 0$.

Здесь $\Delta_{\oplus} > 0$ для $z < n-(n-r-1)/2r$, $\Delta_{\otimes} > 0$ для $z < n-r$.

б) При $\gamma_j = const$:

$\Delta_{\oplus} = r[2n-(r+1)^2]-2zr$,

$\Delta_{\otimes} = (r-1)(n-r-1-z)$, $\Delta_{\copyright} = 0$.

Здесь $\Delta_{\oplus} > 0$ для $z < [2n-(r+1)^2]/4$, $\Delta_{\otimes} > 0$ для $z < n-r-1$.

3) Для параллельно-последовательных алгоритмов.

а) При $\gamma_j \neq const$:

$\Delta_{\oplus} = r[2(n-r)-1]-(2r+1)z$,

$\Delta_{\otimes} = (r-1)^2(n-r-z)$, $\Delta_{\copyright} = 0$.

Здесь $\Delta_{\oplus} > 0$ для $z < r[2(n-r)-1)/(2r+1)$, $\Delta_{\otimes} > 0$ для $z < n-r$.

б) При $\gamma_j = const$:

$\Delta_{\oplus} = r(n-r-1)-(2r+1)z$,

$\Delta_{\otimes} = (r-1)(n-r-2-z)$, $\Delta_{\copyright} = 0$.

Здесь $\Delta_{\oplus} > 0$ для $z < r(n-r-1)]/(2r+1)$, $\Delta_{\otimes} > 0$ для $z < n-r-2$.

Таким образом:

1) при $\gamma_j \neq const$ для большинства практических случаев все алгоритмы перекодирования требуют выполнения меньшего количества операций умножения в конечном поле по сравнению с алгоритмами кодирования;

2) при $\gamma_j \neq const$ и $\gamma_j = const$ для большинства практических случаев последовательный и параллельно-последовательный алгоритмы перекодирования требуют выполнения меньшего количества операций умножения в конечном поле по сравнению с аналогичными алгоритмами кодирования;

3) при $\gamma_j \neq const$ и $\gamma_j = const$ все алгоритмы перекодирования требуют выполнения меньшего количества операций сложения в конечном поле по сравнению с алгоритмами кодирования для определённых значений $z$.

4) при $\gamma_j \neq const$ параллельный алгоритм перекодирования требует выполнения меньшего количества операций умножения в конечном поле по сравнению параллельным алгоритмом кодирования для определённых значений $z$.

*2.3. Сравнение сложностей алгоритмов кодирования неполного кода*

Сравнение параллельного и последовательного алгоритмов показывает:

1) при $L^{(i)}(x) \neq const$:

а) для последовательного алгоритма требуется меньше операций сложения в поле $GF(2^m)$ на величину $k$ и операций умножения на величину $(r-1)^2 k$;

2) при $L^{(i)}(x) = const$:

а) количество операций сложения и умножения в поле $GF(2^m)$ для параллельного алгоритма кодирования меньше на величины $r(r-1)/2$;

б) количество регистров для хранения коэффициентов $L^{(i)}(x)$ меньше для параллельного алгоритма на величину $r(r-1)/2$.

Сравнивая параллельный и параллельно-последовательный алгоритмы, получаем:

1) при $L^{(i)}(x) \neq const$:

а) количество операций умножения в поле $GF(2^m)$ меньше для параллельно-последовательного алгоритма на величину $(r-1)$, но больше операций сложения на величину $(r-1)$;

2) при $L^{(i)}(x) = const$:

а) количество операций сложения и умножения в поле $GF(2^m)$ меньше для параллельного алгоритма на величины $(r-1)$;

б) количество регистров для хранения коэффициентов $L^{(i)}(x)$ меньше для параллельного алгоритма на величину $(r-1)$.

В результате сравнения последовательного и параллельно-последовательного алгоритмов имеем:

1) при $L^{(i)}(x) \neq const$:

а) для последовательного алгоритма требуется меньше операций сложения на величину $(k+r-1)$ и операций умножения на величину $(r-1)[k(r-1)-1]$,

2) При $L^{(i)}(x) = const$:

а) для параллельно-последовательного алгоритма количество операций сложения и умножения в поле $GF(2^m)$ меньше на величины $r(r-3)/2+1$,

б) количество регистров для хранения коэффициентов $L^{(i)}(x)$ меньше для параллельно-последовательного алгоритма на величину $r(r-3)/2+1$.

## Выводы

В результате проведённых исследований получены математические выражения, определяющие сложность алгоритмов кодирования кодов Лагранжа. Выполненный сравнительный анализ этих алгоритмов позволил определить лучшие из них по параметру сложности (количеству операций в конечном поле).

Так сравнение алгоритмов кодирования полных кодов Лагранжа показало:

а) при $L^{(i)}(x) \neq const$ количество операций в конечном поле меньше для последовательного алгоритма кодирования;

б) при $L^{(i)}(x) = const$:

– количество операций в конечном поле и количество регистров для хранения коэффициентов $L^{(i)}(x)$ меньше для параллельно-последовательного алгоритма кодирования,

– при определённых условиях сложность кодирования кода Лагранжа с применением параллельно-последовательного алгоритма меньше сложности самой быстрой процедуры кодирования РС-кодов.

Для неполного кода Лагранжа получены следующие результаты:

а) при $L^{(i)}(x) \neq const$ меньшее количество операций в поле $GF(2^m)$ имеет последовательный алгоритм кодирования;

б) при $L^{(i)}(x) = const$ меньшее количество операций в поле $GF(2^m)$ и количество регистров для хранения коэффициентов $L^{(i)}(x)$ имеет параллельный алгоритм кодирования.

Сравнением алгоритмов перекодирования установлено, что с учётом того, что операция умножения является более сложной по сравнению с операцией сложения, для перекодирования необходимо применять последовательный алгоритм. Для большинства практических случаев перекодирования кодового слова при изменении значений информационных символов все алгоритмы перекодирования требуют выполнения меньшего количества операций умножения в конечном поле по сравнению с алгоритмами кодирования. Это особенно важно для систем реального времени, для которых параметры быстродействия, аппаратурной и алгоритмической сложности играют ключевую роль при функционировании систем.

### Список литературы


1. *Амербаев В. М., Бияшев Р. Г.* Интерполяция и коды, исправляющие ошибки. – В кн.: Теория кодирования и информационное моделирование. – Алма-Ата: Наука, 1973. – С. 51-64.

2. *Кубицкий В. И.* Модификация процедуры кодирования полиномиальными кодами. - Библиографический указатель ВИНИТИ «Депонированные научные работы», №1, 1987. - С. 128. № 422 ГА-86 Деп. от 10.09.86.

3. *Кубицкий В. И.* Кодирование для неполного кода Лагранжа. - Проблеми інформатизації та управління: Збірник наукових праць: Випуск 4 (15). – К.: НАУ, 2005. - С. 118-122.

4. *Кубицкий В. И.* Алгоритм преобразования закодированной информации, хранимой в АС УВД. - Всесоюзная научно-техническая конф. «Научно-технический прогресс и эксплуатация воздушного транспорта». Тезисы докладов. – М., 1990. - С. 130-131.

5. *Кубицкий В. И.* Применение полиномиальных кодов при хранении информации в АС УВД. - Научный вестник ГосНИИ “Аэронавигация”, серия «Проблемы организации воздушного движения. Безопасность полетов». №6. – М.: 2006. - С. 147-153.

6. *Кубицкий В. И.* Методы кодирования массивов информации при передаче и хранении в компьютерных системах. - Сбірник тез ІІІ Міжнародної науково-технічної конференції «Комп’ютерні системи та мережні технології» (CSNT-2010). – К.: Вид-во Нац. авіац. ун-ту «НАУ-друк», 2010. – С. 57.

7. *Miller R. L., Truong T. K., Reed I. S.* Fast algorithm for encoding the (255, 223) Reed-Solomon code over GF($2^8$). – Electronics Letters, vol. 16, №6, 1980. - P. 222-223.

8. *Афанасьев В. Б.* Исследование сложности реализации кодов Рида-Соломона: автореф. дис. на соискание ученой степени канд. тех. наук. – М., 1976. – 19 с.


*Подано до редакції ??.??.12*