УДК 004.31

А.В. Палагин [1], академик НАН Украины,
В.Н. Опанасенко [1], д-р техн. наук, **С.Л. Крывый** [2], д-р физ.-мат. наук
[1] Ин-т кибернетики им. В.М. Глушкова НАН Украины
(Украина, 03680, Киев-187, ГСП, пр-т Академика Глушкова, 40,
тел. (044) 5262598, e-mail: vlopanas@ukr.net),
[2] Киевский национальный университет им. Т.Г. Шевченко
(Украина, 03680, Киев-187, ГСП, пр-т Академика Глушкова, 4Д,
тел. (044) 5223433, e-mail: sl.krivoi@gmail.com)

Метод синтеза структур для преобразований циклического кода на основе программируемой пользователем вентильной матрицы

Обоснована корректность функционирования многоуровневой структуры (последовательного и параллельно-последовательного типов) преобразователя циклических кодовых слов, содержащих группу циклически смежных единиц, на основе логических функций *AND* и *XOR*. Предложенные структуры реализованы в элементном базисе ПЛИС.

Обгрунтовано коректність функціонування багаторівневої структури (послідовного і паралельно-послідовного типів) перетворювача циклічних кодових слів, які мають групу циклічно суміжних одиниць, на основі логічних функцій *AND* та *XOR*. Запропоновані структури реалізовано в елементному базисі ПЛІС.

К л ю ч е в ы е с л о в а: циклический код Хемминга, программируемая пользователем вентильная матрица, многоуровневая структура.

Будем рассматривать свойства циклических кодов Хемминга [1], близких (по расстоянию Хемминга) к заданному эталонному коду, применительно к задачам распознавания образов при построении нейроподобных сетей Хемминга. Коды Хемминга относятся к классу линейных кодов, для которых возможна идентификация ошибок в процессе трансмиссии кода. Циклические коды Хемминга и их преобразования относительно оператора циклического сдвига *R*, операций *AND*, *OR*, *XOR* будем использовать для построения оптимальной вычислительной структуры. Введем необходимые обозначения и определения.

Пусть $X = \{0,1\}$ — двоичный алфавит, а $F_n(X)$ — множество всех слов длины n в алфавите X. Ясно, что $|F_n(X)|=2^n$. Пусть $v = (x_{n-1}\, x_{n-2} \ldots x_1\, x_0) \in$

$\in F_n(X)$, а R — оператор циклического сдвига с шагом 1, определение которого имеет вид

$$\forall v = (x_{n-1}\ x_{n-2}\ ...x_1\ x_0) \in F_n(X)\ R_1(v) = (x_0 x_{n-1}\ x_{n-2}\ ...x_1).$$

Это определение обобщается на произвольный шаг i сдвига очевидным образом:

$$\forall v = (x_{n-1}\ x_{n-2}\ ...x_1\ x_0) \in F_n(X)\ R_i(v) = (x_{i-1}\ ...x_1\ x_0\ x_{n-1}\ ...x_i).$$

Множество V слов из $F_n(X)$, замкнутое относительно оператора сдвига R_i, назовем циклическим кодом Хемминга. Замкнутость множества V означает, что $\forall v \in V(R_i(v) \in V)$. Циклический код Хемминга можно рассматривать как подмножество слов, близких (по расстоянию Хемминга) к заданному эталонному слову в алфавите X. Если в качестве эталонного слова выбирается единичное слово (состоящее из единиц), то в случае расстояния Хемминга, равного единице, все слова из множества V, кроме эталонного, порождаются произвольным его элементом в силу замкнутости этого множества относительно оператора циклического сдвига. Такое слово в дальнейшем будем называть порождающим словом.

Введем следующие обозначения:

$H_m(1)$ — циклический код длиной n, порождающее слово которого включает m-компонентную $m = 1 \div (n-1)$ группу циклически смежных единиц и d-компонентную $d = (n-m)$ группу циклически смежных нулей;

U — циклический код длиной n, порождающее слово которого включает n-компонентную группу циклически смежных единиц.

Постановка задачи. Пусть имеется полное множество G кодовых слов длиной n и циклический код $H_m(1) \in G$, порождающее слово которого имеет группу циклически смежных единиц длиной m. Необходимо построить структуру, реализующую отображение $\Im$, определяемое в виде $\Im(H_m(1)) = 1, \Im(G \setminus H_m(1)) = 0$.

Задача может быть решена путем последовательного соединения [2, 3] циклических структур типа *AND*, имеющих n входов и n выходов, а также структуры *OR*, имеющей n входов и один выход. Структуры имеют r-уровневую организацию, их s-й уровень ($s = 1 \div r$) содержит n логических элементов *AND*, реализующих преобразование $f(a_i, a_{(i+j_s) \bmod(n)})$, $\forall i = 0 \div (n-1), 1 \le j \le (n-1)$, где f — логическая функция *AND* двух переменных; $a_i, a_{(i+j_s) \bmod(n)}$ — однобитовые компоненты входного двоичного слова; $j_s \in [1, (n-2)]$ — шаг циклического сдвига. Все элементы одного уровня настраиваются на выполнение одной и той же логической функции *AND*.

Рассмотрим два варианта структурной реализации поставленной задачи.

Варианты структурной реализации. ***Последовательная структура*** **(ПС).** Согласно таблице истинности логической функции *AND* и струк-

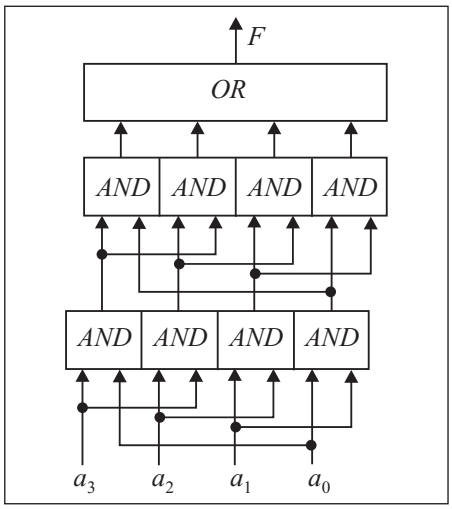

Рис. 1. Схема ПС преобразователя циклических кодов на основе операции *AND*

туры связей ($\forall s$, $j_s = 1$) мощность группы циклически смежных единиц при увеличении номера уровня s уменьшается на одну единицу. Таким образом, для преобразования кодового слова с заданным значением m необходимо иметь $s = (m-1)$ уровней. В общем случае это вытекает из следующей теоремы.

Теорема 1. Если на вход s-го уровня $(1 \le s < r)$ структуры с такими связями подается слово, содержащее группу из d циклически смежных нулей, то на его выходе будет слово, содержащее группу из $(d+1)$ циклически смежных нулей.

Доказательство методом математической индукции по числу уровней s.

Базис индукции $s = 1$. В этом случае на вход первого уровня подается слово, содержащее только один нуль. Пусть этот нуль соответствует i-му входу, т.е. $x_i = 0$. Поскольку x_i на выходе участвует в формировании только двух произведений, $(x_{i-1}\, x_i)$ и $(x_i\, x_{i+1})$, на выходе первого уровня будет только два нуля. Эти два нуля будут циклически смежными, т.е. появятся на циклически смежных выходах первого уровня.

Шаг индукции. Пусть утверждение теоремы 1 верно для уровня $s > 1$, т.е. на вход уровня s подается циклическое слово, содержащее группу из $d > 1$ циклически смежных нулей. Пусть это будут $a^s_{i+1} a^s_{i+2} \ldots a^s_{i+d}$. Тогда на входе s-го уровня получаем группу $a^s_i a^s_{i+1}$, $a^s_{i+1} a^s_{i+2}$, ..., $a^s_{i+d} a^s_{i+d+1}$ циклически смежных нулей, число которых будет $(d+1)$. Теорема доказана.

Следствие 1. Для выбранной структуры реализации преобразователя значение 1 на выходе будет только тогда, когда на ее вход подается циклическое кодовое слово с расстоянием Хемминга, равным единице относительно эталонного слова, состоящего из единиц.

Действительно, в этом случае на выходе $(m-1)$-го уровня будет находится только одна единица, чем гарантировано выходное значение 1 для всего подмножества $H_m(1)$.

Пример 1. Схема реализации ПС при $n = 4$, $j_s = 1$ приведена на рис. 1, а результаты поуровневого преобразования — в табл. 1.

Параллельно-последовательная структура **(ППС)** может быть реализована на уровнях $r = \log_2 n$, а все элементы настроены на реализацию логической функции *AND*. Каждый уровень параллельно-последовательного преобразователя имеет регулярную структуру связей, т.е. на вход любого *i*-го логического элемента уровня *s* поступают компоненты с индексами *i* и $(i+s)$ (значение $i + s$ берется по модулю *n*).

Пусть задана структура, изображенная на рис. 2. Опишем зависимость между разрядностью *n* входного слова, числом уровней *r* и числом циклических смежных нулей *d* в этом входном слове, которые гарантируют выходное значение 1 всей структуры в целом. Эта зависимость вытекает из следующей теоремы.

Теорема 2. Переменные *n*, *r* и *d* для данной структуры связаны зависимостью $d = (n-1)-(r(r+1)/2)$.

Доказательство. Пусть входное слово длины *n* имеет *d* циклически смежных нулей. Тогда в силу теоремы 1 на выходе первого уровня получим слово, содержащее $(d+1)$ циклически смежных нулей. На выходе второго уровня получим слово q_1, являющееся результатом покомпонент-

Таблица 1

Входной код				Значения на выходе								Выход *F*
				1-го уровня				2-го уровня				
a_3	a_2	a_1	a_0	a_3	a_2	a_1	a_0	a_3	a_2	a_1	a_0	
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	0	0	1	1	0	0	0	1	
1	0	1	1	1	0	0	1	1	0	0	0	
1	1	0	1	1	1	0	0	0	1	0	0	
1	1	1	0	0	1	1	0	0	0	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	
0	0	1	0	0	0	0	0	0	0	0	0	
0	0	1	1	0	0	0	1	0	0	0	0	
0	1	0	0	0	0	0	0	0	0	0	0	
0	1	0	1	0	0	0	0	0	0	0	0	
0	1	1	0	0	0	1	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	1	1	0	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	0	0	
1	1	0	0	0	1	0	0	0	0	0	0	

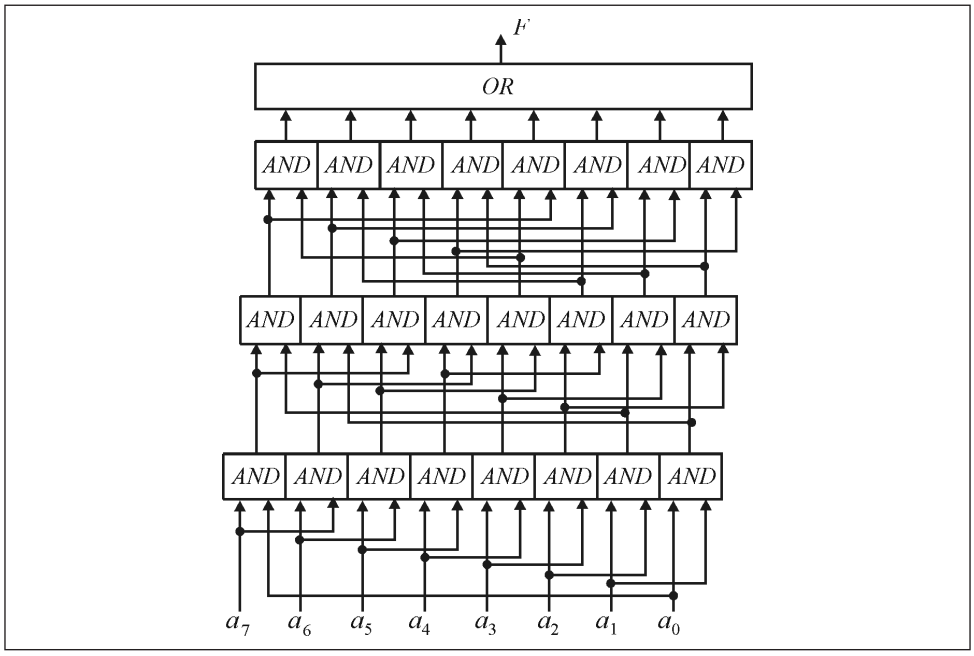

Рис. 2. Схема ППС преобразователя циклических кодов на основе операции *AND*

ной конъюнкции выходного слова первого уровня с его циклической перестановкой вида $R(R(q_1))$. В этом слове группа нулей сместится на два шага вправо, в результате чего получим слово q_2, содержащее $(d+1)+2$ нулей. Далее по индукции на выходе третьего уровня получаем слово q_3, содержащее $(d+1)+2+3$ нулей, $(d+1)+2+3+...+r$ нулей и так далее, т.е. $d^1 = d+(r(r+1)/2)$. Гарантией того, что на выходе всей структуры, т.е. при выполнении последней операции *OR*, будет единица, является зависимость $d^1 = n-1$. Это означает, что на входе операции *OR* будет одна единица. Следовательно, $d=(n-1)-(r(r+1)/2)$, что и требовалось доказать.

Из теоремы 2 вытекает, что не любое преобразование циклического кода Хемминга реализуемо такой структурой при заданных значениях n и d. Например, если $n=20$ и $d=1$, то $r(r+1)=36$. Это уравнение не имеет решения в целых натуральных числах. Оно имеет решение при $d=4$. Поскольку $d=(20-1)-(r(r+1)/2)$, то $r(r+1)=38-8=30=5\times6$. Следовательно, $r=5$.

Это свидетельствует также о том, что при фиксированных n и r можно разделить множество циклических кодов Хемминга, найдя число d. Действительно, пусть $n=16$ и $r=4$. Тогда $d=(16-1)-(4\times5)/2=15-10=5$. Следовательно, все входные слова, содержащие не более пяти циклически смежных нулей (расстояние Хемминга ≤ 5 от эталонного слова $H(1)$), гарантируют значение 1 на выходе структуры в целом.

Пример 2. Пусть входное слово имеет вид $q_0 = 0011111111111100$. Это слово покомпонентно умножается на слово 0001111111111110, что формирует на выходе первого уровня слово $q_1 = 0001111111111100$.

Затем это слово покомпонентно умножается на слово 0000011111111111, что формирует на выходе второго уровня слово $q_2 = 0000011111111100$.

Далее это слово покомпонентно умножается на слово 1000000011111111, что формирует на выходе третьего уровня слово $q_3 = 0000000011111100$.

Это слово покомпонентно умножается на слово 1100000000001111, в результате чего на выходе четвертого уровня формируется слово $q_4 = 0000000000001100$.

Это означает, что на выходе структуры будет значение 1.

В данной структуре $n=16$, $d=4$, $r=4$. Тогда на выходе будет $d=(n-x)-(r(r+1)/2)$, т.е. $4=(16-x)-10$ или $x=6-4=2$, т.е. на выходе r-го уровня будет две единицы, что гарантирует выход единицы на всей структуре. Из изложенного следует, что при заданных значениях n, d или n, r, или r, d можно найти r, d или n.

Рассмотрим теперь, на какой структуре реализуется преобразование циклического кода, когда $n=2^r$, $r=\log_2 n$, а $d=1$. Проведенный анализ рассмотренной выше структуры позволяет определить структуру, на которой можно реализовать любое преобразование с этими параметрами. В такой структуре на уровне $1 \le s \le r-1$ сдвиг выполняется с шагом 2^{s-1}, а на предпоследнем уровне r — с шагом $(2^{s-1}-1)$.

Сформулируем следующую теорему.

Теорема 3. Структура, реализующая данное преобразование, связывает значения n, r и d так:

$$d^1 = d+(2^{r-1}-1)+2^{r-1}-1 = d+2^r-2 = 2^r-1,$$

где d^1 — число циклически смежных нулей на выходе r-го уровня.

Доказательство. Если реализация данной структуры выполняется на уровне $1 \le s \le r-1$ с шагом 2^{s-1}, то на уровне $s=1$ шаг будет $2^0=1$, на уровне $s=2$ шаг будет $2^1=2$, на третьем уровне — $2^2=4$ и на $(r-1)$-м уровне — 2^{r-2}. Тогда на r-м уровне будет

$$d+(2^{r-1}-1)+2^{r-1}-1 = d+2^r-2 = 2^r-1.$$

Действительно, на выходе первого уровня (согласно теореме 1) будем иметь два циклически смежных нуля. На втором уровне, в силу того что шаг сдвига равен двум, получаем на выходе слово с четырьмя циклически смежными нулями, на третьем — с восемью циклически смежными нулями и так далее. На $(r-1)$-м уровне получаем на выходе слово с 2^{r-1} цикли-

чески смежными нулями. На r-м уровне к 2^{r-1} нулям добавляется $2^{r-1} - 1$ нулей согласно структуре связей. Тогда получаем

$$d^1 = d + (1 + 2 + ... + 2^{r-2}) + 2^{r-1} - 1 = d + 2^r - 2 = 2^r - 1,$$

что и требовалось доказать. Из этой теоремы следует, что на выходе всей структуры в целом будет получено значение 1.

Пример 3. Пусть $n = 2^4 = 16$, $r = 4$, $d = 1$. На вход первого уровня поступает слово, например, $q_0 = 1011111111111111$. Это слово умножается на слово с циклическим сдвигом на единицу (1101111111111111), что дает на выходе первого уровня слово $q_1 = 1001111111111111$.

Это слово умножается на слово с циклическим сдвигом на два (1110011111111111), в результате чего на выходе второго уровня получаем слово $q_2 = 1000011111111111$.

Это слово умножается на слово с циклическим сдвигом на четыре (1111100001111111), в результате чего на выходе третьего уровня получаем слово $q_3 = 1000000001111111$.

Это слово умножается на слово с циклическим сдвигом на семь (1111111100000000),что дает на выходе четвертого (последнего) уровня слово $q_4 = 1000000000000000$.

Результаты поуровневого преобразования для ППС ($n = 8$) приведены в табл. 2.

Циклический код, содержащий два нуля ($s = 2$). Слово представлено сочетанием групп циклически смежных единиц (где $\underline{d_1}$ — группа циклически смежных единиц максимальной длины $\forall i = \overline{1-(s-1)}$, при этом $|d_1| \geq |d_2| \geq ... \geq |d_i| \geq |d_{s-1}|$) и групп циклически смежных нулей. Число групп зависит от расстояния Хемминга (s).

Число уровней структуры, настроенных на реализацию логической функции *AND*, определяется величиной $r = (d_1 - 1)$. Остальные уровни структуры настраиваются на реализацию логической функции A, т.е. функции транзита значения переменной, поступающей на вход A двухвходового логического элемента. Таким образом, на выходе получаем слово, содержащее только одну единицу.

Пример 4. Пусть $n = 16$. Число уровней для получения на выходе циклического кода с единственной единицей определяется величиной $r = (d_1 - 1)$ (табл. 3).

Рассмотрим ПС и ППС (рис. 3), базовой операцией которых является операция $\oplus$ (*XOR*). Число уровней структуры, настраиваемых на реализацию логической функции *XOR*, определяется соответствующими правилами в зависимости от числа нулей и их местоположения во входном слове.

Таблица 2

Преобразование циклических кодов для ППС							
a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
Входной код							
1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	0
На выходе 1-го уровня							
1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1
1	0	0	1	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	0	0	1	1	1
1	1	1	1	0	0	1	1
1	1	1	1	1	0	0	1
1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	0
На выходе 2-го уровня							
1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1
1	0	0	0	0	1	1	1
1	1	0	0	0	0	1	1
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0
На выходе 3-го уровня							
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0

Таблица 3

Выход на уровне	Входной код 1110 0111 1111 1111	Входной код 1110 1111 1111 1110
	$d_1 = 14, d_2 = 0$	$d_1 = 11, d_2 = 3$
1	1110 0011 1111 1111	0110 0111 1111 1110
2	1110 0001 1111 1111	0010 0011 1111 1110
3	1110 0000 1111 1111	0000 0001 1111 1110
4	1110 0000 0111 1111	0000 0000 1111 1110
5	1110 0000 0011 1111	0000 0000 0111 1110
6	1110 0000 0001 1111	0000 0000 0011 1110
7	1110 0000 0000 1111	0000 0000 0001 1110
8	1110 0000 0000 0111	0000 0000 0000 1110
9	1110 0000 0000 0011	0000 0000 0000 0110
10	1110 0000 0000 0001	0000 0000 0000 0010
11	1110 0000 0000 0000	
12	0110 0000 0000 0000	
13	0010 0000 0000 0000	
Число уровней	$r=(d_1-1)=14-1=13$	$r=(d_1-1)=11-1=10$

На выходе таких структур (ПС и ППС) получаем слово V = 1010 ... 1010 либо V_1 = 0101 ... 0101. Далее с помощью оператора циклического сдвига R_1 и операции *XOR* получаем единичное слово U. Для формирования выхода структуры F применяем n-входовой логический элемент *AND*.

Свойства оператора циклического сдвига относительно операции *XOR*. Среди слов в множестве $F_n(X)$ выделим специальные слова:

- нулевое слово $Z = 000...000 \in F_n(X)$;
- единичное слово $U = 111...111 \in F_n(X)$ (эталон);
- $V = 1010...1010, V_1 = 0101...0101 \in F_n(X)$;
- канонические слова $e_1 = 100...000, e_2 = 0100...000, e_n = 000...001 \in F_n(X)$.

Название «канонические слова» обусловлено тем, что любое слово $J \in F_n(X)$ представляется в виде линейной комбинации канонических слов, т.е. $\forall J \in F_n(X)$, $J = c_1e_1 \oplus c_2e_2 \oplus ... \oplus c_ne_n \oplus$, где $c_i \in \{0,1\}$ — коэффициенты разложения.

Для произвольного слова $J \in F_n(X)$ введем слово $\overline{J} \in F_n(X)$, которое определяется так: если $J = a_{n-1}a_{n-2} ... a_1a_0$, то $\overline{J} = \overline{a_{n-1}a_{n-2} ... a_1a_0}$, где

$$\overline{a}_i = \begin{cases} 0 \text{ при } a_i = 1, \\ 1 \text{ при } a_i = 0. \end{cases}$$

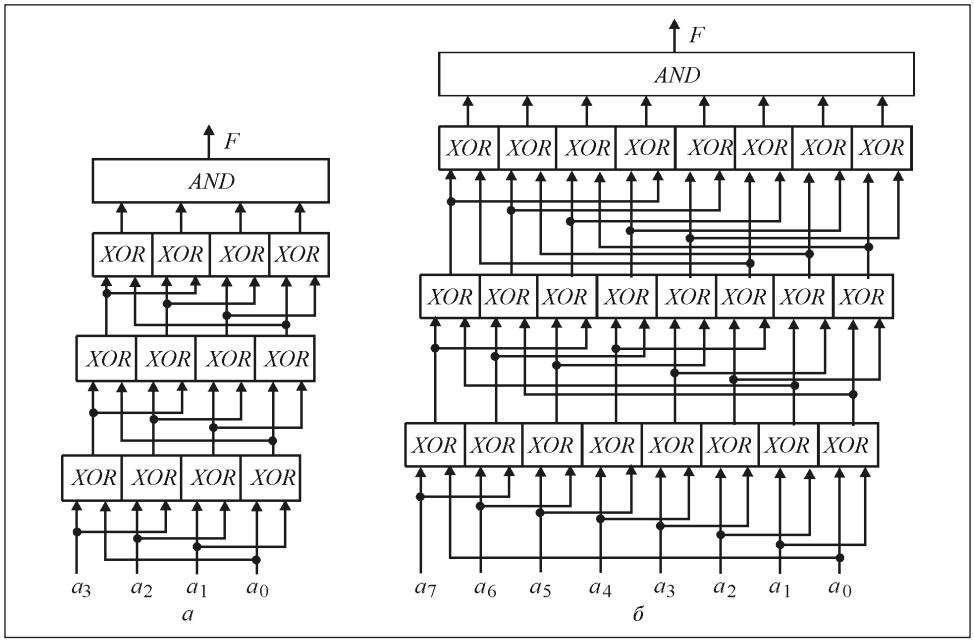

Рис. 3. Схемы ПС (*а*) и ППС (*б*) преобразователя циклических кодов на основе операции *XOR*

Из этого определения вытекают следующие свойства введенных слов:

$$V \oplus V_1 = U,\ V = \overline{V}_1,\ V_1 = \overline{V},\ V \oplus \overline{V}_1 = Z = \overline{V} \oplus V_1.$$

Напомним, что оператором циклического сдвига слова J с шагом i называется преобразование вида

$$R_i(J) = a_{i-1}a_{i-2} \ldots a_0 a_{n-1}a_{n-2} \ldots a_i. \qquad (1)$$

Из этого определения вытекают следующие очевидные свойства:

$$R_n(J) = R_0(J) = J,\ \ R_i(Z) = Z,\ \ R_i(U) = U,\ \ R_1(V) = V_1,\ \ R_1(V_1) = V,$$
$$U \oplus R_i(J) = Z,\ \ V \oplus R_1(V) = U,\ \ V_1 \oplus R_1(V) = U, \qquad (2)$$
$$R_{2r}(V) = V,\ \ R_{2r}(V_1) = V_1,\ \ R_{2r+1}(V_1) = V,\ \ \overline{J} = U \oplus J.$$

Далее, $e_1 = R_0(e_1) = R_n(e_1)$, $e_2 = R_1(e_1)$, ..., $e_n = R_{n-1}(e_1)$, где $e_1 = 100 \ldots 000$ — каноническое слово.

Пример 5. Слово $J = 01100101 \in F_8(X)$ в таком представлении имеет вид $J = R_1(e_1) \oplus R_2(e_1) \oplus R_5(e_1) \oplus R_7(e_1)$. Конец примера.

Основные свойства оператора сдвига и операции $\oplus$ сформулированы в следующей теореме.

Теорема 4. Для любых $i, j \in \overline{1, n}$, $J \in F_n(X)$

а) $R_i(R_j(J)) = R_j(R_i(J)) = R_{i+j}(J) = R_p(J)$, где $p = i + j \pmod n$;

б) $R_1(J + R_i(J)) = R_1(J) \oplus (R_{i+1}(J)) = R_p(J)$, где $p = i + 1 \pmod n$;

в) $J \oplus R_i(J) = \bar{J} \oplus R_i(\bar{J})$, где $\bar{J}$ — дополнение J.

Д о к а з а т е л ь с т в о. а) Если $i + j < n$, то согласно определению оператора циклического сдвига получаем

$$R_i(J) = a_{i-1}a_{i-2} \ldots a_0 a_{n-1}a_{n-2} \ldots a_i,\ R_j(J) = a_{j-1}a_{j-2} \ldots a_0 a_{n-1}a_{n-2} \ldots a_j,$$

$$R_i(R_j(J)) = a_{i+j-1}a_{i+j-2} \ldots a_{i+j} \text{ и } R_j(R_i(J)) = a_{i+j-1}a_{i+j-2} \ldots a_{i+j}.$$

Отсюда вытекает равенство $R_i(R_j(J)) = R_j(R_i(J)) = R_{i+j}(J)$. Если $i + j \geq n$, то $i + j = n + p$, где $p = i + j \pmod n$. Тогда, в силу доказанного выше, можно записать $R_{i+j}(J) = R_n(R_p(J)) = R_p(J))$ с учетом свойств (2) для R_n.

б) В результате непосредственных вычислений получаем $J \oplus R_i(J) = a^1_{n-1}a^1_{n-2} \ldots a^1_1 a^1_0$, где

$$a^1_j = \begin{cases} a_j \oplus a_{j+i-n}, j > (n-i-1), \\ a_j \oplus a_{j+i(\bmod n)}, j \leq (n-i-1), \end{cases}$$

$$R_1(J \oplus R_i(J)) = a^1_0 a^1_{n-1}a^1_{n-2} \ldots a^1_1,$$

$$R_1(J \oplus R_{i+1}(J)) = a_0 a_{n-1} \ldots a_1 \oplus a_i a_{i-1} \ldots a_0 a_{n-1} \ldots a_{i+1} = a^1_0 a^1_{n-1}a^1_{n-2} \ldots a^1_1.$$

Отсюда следует требуемое равенство $R_1(J \oplus R_i(J)) = R_1(J \oplus R_{i+1}(J))$, если $i < n - 1$. Если $i = n - 1$, то $R_1(J \oplus R_i(J)) = R_1(J) \oplus J$, что можно проверить непосредственно.

в) Если $J = a_{n-1}a_{n-2} \ldots a_1 a_0$, то $\bar{J} = U \oplus J$. В силу доказанных свойств а) и б), а также свойств (2), получаем

$$\bar{J} \oplus R_i(\bar{J}) = U \oplus J \oplus R_i(U \oplus J) = U \oplus J \oplus R_i(U) \oplus R_i(J) =$$

$$= U \oplus J \oplus U \oplus R_i(J) = J \oplus R_i(J).$$

Теорема 4 доказана.

Доказанные свойства оператора циклического сдвига R_i и операции $\oplus$ позволяют оптимизировать выражения, получаемые в процессе вычислений, а также и вычислительную структуру.

Пример 6. Выражение $J \oplus R_1(J) \oplus R_1(J \oplus R_1(J))$ редуцируется к виду $J \oplus R_2(J)$;

$$J \oplus R_1(J) \oplus R_2(J \oplus R_1(J)) \oplus R_1(J \oplus R_1(J) \oplus R_2(J \oplus R_1(J))) = J \oplus R_4(J).$$

Конец примера.

Из простейших свойств (2) и теоремы 4 следует, что любое входное слово $J \in F_n(X)$ с помощью оператора циклического сдвига R_i и операции $\oplus$ редуцируется к одному из слов U, Z, V и V_1. Рассмотрим случай редукции

к словам V и V_1, так как слова U и Z не несут никакой информации о входном слове J.

Входное слово с одним нулем (расстояние Хемминга равно единице). В силу свойства в) теоремы 4 применение редукции к слову, например $J = 011...111$, эквивалентно применению этих редукций к слову $e_1 = \bar{J} = 100...000$.

Теорема 5. Если входное слово вычислительной структуры содержит только один нуль, то для редукции его к слову V (соответственно V_1) необходимо $\log_2 n - 1$ уровней этой структуры.

Д о к а з а т е л ь с т в о. Рассмотрим частный случай входного слова $e_1 = 100...000$. Тогда утверждение теоремы становится очевидным, поскольку

$$e_1 \oplus R_2(e_1) = 1010...000,$$

$$e_1 \oplus R_2(e_1) \oplus R_4(e_1 \oplus R_2(e_1)) = e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1)),$$

$$e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1) \oplus R_8(e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1)) =$$

$$= e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1) \oplus R_8(e_1) \oplus R_{10}(e_1) \oplus R_{12}(e_1) \oplus R_{14}(e_1).$$

Очевидное продолжение в конце приводит к слову V.

Общий случай сводится к рассмотренному частному следующим образом. Пусть $e_i = \bar{J}$, при этом J — входное слово. Тогда $e_i = R_{i-1}(e_1)$. Отсюда следует, что необходимо выполнить редукции над словом e_1, а затем к результату применить оператор R_{i-1}. Однако в силу свойства а) теоремы 4 это равносильно применению операторов R_2, R_4, R_8 и так далее к слову $R_{i-1}(e_1)$. Действительно,

$$e_i \oplus R_2(e_i) = R_{i-1}(e_1) \oplus R_{i+1}(e_1) = R_{i-1}(e_1 \oplus R_2(e_1)),$$

$$e_i \oplus R_2(e_i) \oplus R_4(e_i \oplus R_2(e_i)) = R_{i-1}(e_1 \oplus R_2(e_1)) \oplus R_{i-1}(R_4(e_1) \oplus R_6(e_1)) =$$

$$= R_{i-1}(e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1)) =$$

$$= R_{i-1}(e_1) \oplus R_{i+1}(e_1) \oplus R_{i+3}(e_1) \oplus R_{i+5}(e_1)).$$

Таким образом, и в этом случае вычислительная структура должна иметь $\log_2 n - 1$ уровней. Теорема доказана.

Следствие 2. Если e_i имеет нечетный номер i (начиная с нуля и считая символы слева направо), то слово e_i редуцируется к слову V, иначе — к слову V_1.

Действительно, применение оператора сдвига R_{i-1} и операции $\oplus$ не изменяет позиции единицы, находящейся на i-м месте в слове e_i. Это следствие подтверждается данными для входных слов из $F_8(X)$, приведенными в табл. 4 и 5, и для входных слов из $F_{16}(X)$ — данными,

приведенными в табл. 6. На рис. 4 представлены ППС, построенные с использованием различных операторов сдвига.

Входное слово с циклически смежными нулями. Из свойства в) теоремы 4 вытекает следующее обстоятельство. Если $n = 2^k$, то рассматривать нужно только 2^{k-1} входных слов, т.е. ровно половину, поскольку для второй половины ситуация повторяется.

Обоснование общего случая для входного слова с циклически смежными нулями вытекает из следующей теоремы.

Теорема 6. Если входное слово вычислительной структуры содержит $r > 1$ циклически смежных нулей, то для редукции его к слову V (или V_1) требуется $\log_2 n - 1$ уровней этой структуры.

Доказательство. Рассмотрим последовательно случаи $r = 2$ и $r = 3$. Используя свойство в) теоремы 4, можем записать для $r = 2$

$$J = R_i(e_1) \oplus R_{i+1}(e_1) = R_i(e_1 \oplus R_1(e_1))$$

для $r = 3$

$$J = R_i(e_1) \oplus R_{i+1}(e_1) \oplus R_{i+2}(e_1) = R_i(e_1 \oplus R_1(e_1) \oplus R_2(e_1)).$$

Как видим, выполняется редукция слов $e_1 \oplus R_1(e_1)$ и $e_1 \oplus R_1(e_1) \oplus R_2(e_1)$.

Таблица 4

Выход на уровне	ПС	ППС	ПС	ППС
	Входной код 0111 1111		*Входной код* 1011 1111	
1	1100 0000	1010 0000	0110 0000	0101 0000
2	1010 0000	1010 1010	0101 0000	0101 0101
3	1111 0000	1111 1111	0111 1000	1111 1111
4	1000 1000		0100 0100	
5	1100 1100		0110 0110	
6	1010 1010		0101 0101	
7	1111 1111		1111 1111	
	Входной код 1111 0111		*Входной код* 1111 1011	
1	0000 1100	0000 1010	0000 0110	0000 0101
2	0000 1010	1010 1010	0000 0101	0101 0101
3	0000 1111	1111 1111	1000 0111	1111 1111
4	1000 1000		0100 0100	
5	1100 1100		0110 0110	
6	1010 1010		0101 0101	
7	1111 1111		1111 0111	

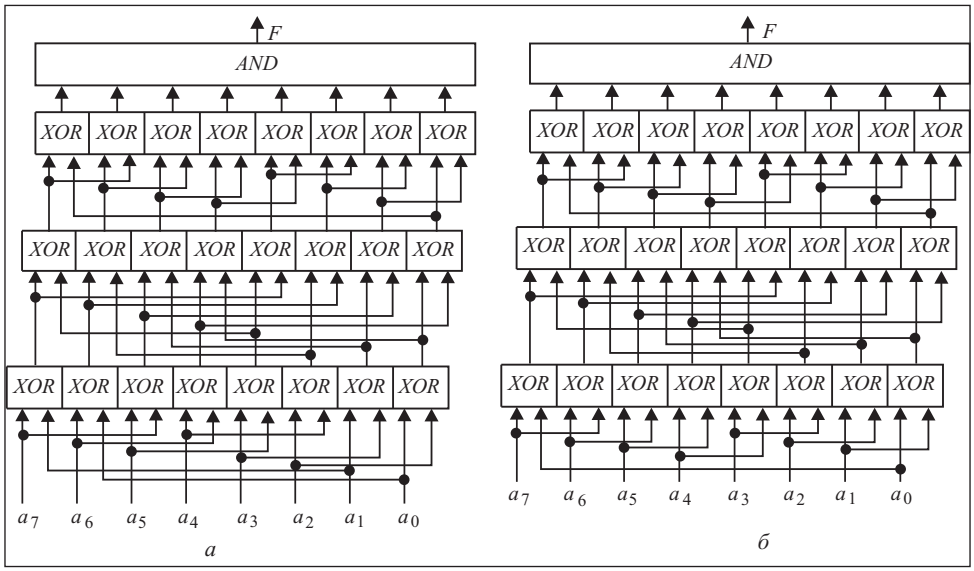

Рис. 4. Схемы ППС преобразователей, построенные на основе операторов сдвига R_2, R_4 (*а*) и R_1, R_4 (*б*)

Таблица 5

Выход на уровне	ПС	ППС	ПС	ППС
	Входной код 0011 1111		*Входной код* 1001 1111	
1	1010 0000	1010 0000	0101 0000	0101 0000
2	1111 0000	1010 1010	0111 1000	0101 0101
3	1000 1000	1111 1111	0100 0100	1111 1111
4	1100 1100		0110 0110	
5	1010 1010		0101 0101	
6	1111 1111		1111 1111	
	Входной код 1111 0011		*Входной код* 1111 1001	
1	0000 1010	0000 1010	0000 0101	0000 0101
2	0000 1111	1010 1010	1000 0111	0101 0101
3	1000 1000	1111 1111	0100 0100	1111 1111
4	1100 1100		0110 0110	
5	1010 1010		0101 0101	
6	1111 1111		1111 1111	

В случае $r=2$, применяя оператор R_1 к $e_1 \oplus R_1(e_1)$, получаем слово вида $e_1 \oplus R_2(e_1)$, рассмотренное в теореме 5. Для его редукции с помощью операторов R_4, R_8 (пропуская R_2) требуется $\log_2 n-2$ уровня, а с учетом первой редукции (с помощью R_1) — $\log_2 n-1$ уровней.

В случае $r=3$ слово $e_1 \oplus R_1(e_1) \oplus R_2(e_1)$ преобразуем с помощью операторов R_2, R_4, R_8 (и так далее в случае необходимости):

$$e_1 \oplus R_1(e_1) \oplus R_2(e_1) \oplus R_2(e_1 \oplus R_1(e_1) \oplus R_2(e_1)) =$$
$$= e_1 \oplus R_1(e_1) \oplus R_2(e_1) \oplus R_2(e_1) \oplus R_3(e_1) \oplus R_4(e_1) =$$
$$= e_1 \oplus R_1(e_1) \oplus R_3(e_1) \oplus R_4(e_1).$$

Далее,

$$e_1 \oplus R_1(e_1) \oplus R_3(e_1) \oplus R_4(e_1) \oplus R_4(e_1 \oplus R_1(e_1) \oplus R_3(e_1) \oplus R_4(e_1)) =$$
$$= e_1 \oplus R_1(e_1) \oplus R_3(e_1) \oplus R_4(e_1) \oplus R_4(e_1) \oplus R_5(e_1) \oplus R_7(e_1) \oplus R_8(e_1)) =$$
$$= e_1 \oplus R_1(e_1) \oplus R_3(e_1) \oplus R_5(e_1) \oplus R_7(e_1) \oplus R_8(e_1).$$

Если $n = 8$, то это слово имеет вид V_1, а окончательное слово — вид $R_i(V_1)$. Если $n > 8$, то применяем оператор R_8 и так далее. В любом случае первый и последний (как и средние) члены, уничтожаются и на выходе получаем $R_i(V)$ или $R_i(V_1)$. При этом число уровней будет $\log_2 n-1$.

Таблица 6

Выход на уровне	Входной код 1111 1111 1111 1101	
	ПС	ППС
1	0000 0000 0000 0011	1000 0000 0000 0010
2	1000 0000 0000 0010	1010 1000 0000 0010
3	1100 0000 0000 0011	1010 1010 1010 1010
4	0010 0000 0000 0010	1111 1111 1111 1111
5	0011 0000 0000 0011	
6	1010 1000 0000 0010	
7	1111 1100 0000 0011	
8	0000 0010 0000 0010	
9	0000 0011 0000 0011	
10	1000 0010 1000 0010	
11	1100 0011 1100 0011	
12	0010 0010 0010 0010	
13	0011 0011 0011 0011	
14	1010 1010 1010 1010	
15	1111 1111 1111 1111	

Теперь рассмотрим последовательно случаи $r=5,6,7$ (в случае $r=4$ ситуация подобна случаю $r=2$). Входное слово посредством применения оператора R_1 преобразуется к виду $J \oplus R_i(J) = R_i(e_1) \oplus R_{i+4}(e_1)$. Далее применяем операторы R_2, R_8 (пропуская R_4) и так далее. Если $r=5$, то по аналогии с $r=3$ применяем операторы R_2, R_4, R_8 (и так далее в случае необходимости). Действительно, если $n=16$, то

$$e_1 \oplus R_1(e_1) \oplus R_2(e_1) \oplus R_3(e_1) \oplus R_4(e_1) \oplus$$
$$\oplus R_2(e_1 \oplus R_1(e_1) \oplus R_2(e_1) \oplus R_3(e_1) \oplus R_4(e_1)) =$$
$$= e_1 \oplus R_1(e_1) \oplus R_5(e_1) \oplus R_6(e_1).$$

Далее,

$$e_1 \oplus R_1(e_1) \oplus R_5(e_1) \oplus R_6(e_1) \oplus R_4(e_1) \oplus R_5(e_1) \oplus R_9(e_1) \oplus R_{10}(e_1) =$$
$$= e_1 \oplus R_1(e_1) \oplus R_4(e_1) \oplus R_6(e_1) \oplus R_9(e_1) \oplus R_{10}(e_1).$$

И, наконец,

$$e_1 \oplus R_1(e_1) \oplus R_4(e_1) \oplus R_6(e_1) \oplus R_9(e_1) \oplus R_{10}(e_1) \oplus R_8(e_1) \oplus R_9(e_1) \oplus$$
$$\oplus R_{12}(e_1) \oplus R_{14}(e_1) \oplus R_1(e_1) \oplus R_2(e_1) =$$
$$= e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1) \oplus R_8(e_1) \oplus R_{10}(e_1) \oplus R_{12}(e_1) \oplus R_{14}(e_1) = V.$$

Если $r=6$, то слово $e_1 \oplus R_1(e_1) \oplus R_2(e_1) \oplus R_3(e_1) \oplus R_4(e_1) \oplus R_5(e_1)$ посредством применения оператора R_1 преобразуется к слову $e_1 \oplus R_6$. Далее выполняем преобразования с помощью операторов R_4, R_8 (пропуская R_2) и так далее в случае необходимости. Например, при $n=8$ получаем

$$e_1 \oplus R_6(e_1) \oplus R_4(e_1 \oplus R_6(e_1)) = e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1)) = V.$$

При $n=16$ получаем слово

$$e_1 \oplus R_4(e_1) \oplus R_6(e_1) \oplus R_{10}(e_1) \oplus R_8(e_1) \oplus R_{12}(e_1) \oplus R_{14}(e_1) \oplus R_2(e_1) =$$
$$= e_1 \oplus R_2(e_1) \oplus R_4(e_1) \oplus R_6(e_1) \oplus R_8(e_1) \oplus R_{10}(e_1) \oplus R_{12}(e_1) \oplus R_{14}(e_1) = V.$$

Если $r=7$, то по аналогии с $r=3,5$ применим операторы R_2, R_4, R_8 (и так далее в случае необходимости). Действительно, после применения R_2 к $e_1 \oplus R_1(e_1) \oplus \ldots \oplus R_6(e_1)$ получим слово $e_1 \oplus R_1(e_1) \oplus R_7(e_1) \oplus R_8(e_1)$.

После применения R_4, получаем слово

$$e_1 \oplus R_1(e_1) \oplus R_4(e_1) \oplus R_5(e_1) \oplus R_7(e_1) \oplus R_8(e_1) \oplus R_{11}(e_1) \oplus R_{12}(e_1),$$

а после применения R_8 находим

$$e_1 \oplus R_1(e_1) \oplus R_4(e_1) \oplus R_5(e_1) \oplus R_7(e_1) \oplus R_8(e_1) \oplus R_{11}(e_1) \oplus R_{12}(e_1) \oplus$$
$$\oplus R_8(e_1) \oplus R_9(e_1) \oplus R_{12}(e_1) \oplus R_{13}(e_1) \oplus R_{15}(e_1) \oplus R_{16}(e_1) \oplus R_{19}(e_1) \oplus R_{20}(e_1).$$

В случае $n=16$ это слово равно V_1. Далее метод получения слова V или V_1 становится ясным:

а) если r — четное, то применяем операторы R_1, R_4, R_8 и так далее;

б) если $r=2^m$, то применяем R_1, R_2, R_4, ..., $R_{2^{m-1}}$, $R_{2^{m+1}}$ и так далее;

в) если r — нечетное, то применяем операторы R_2, R_4, R_8 и так далее.

Теорема доказана.

Следствие 3. Если

а) i — четное, а r — нечетное, то входное слово J редуцируется к слову V;

б) i — четное и r — четное, то входное слово J редуцируется к слову V_1;

в) i — нечетное и r — нечетное, то входное слово J редуцируется к слову V;

г) i — нечетное и r — нечетное, то входное слово J редуцируется к слову V_1.

Данные, подтверждающие теорему 6 и следствие 3 для $n=8$, $r=3$ и $n=8$, $r=5$, приведены в табл. 5 и 7, а ППС для их реализации — на рис. 4, *а*.

Входное слово с двумя несмежными нулями. Рассмотрим сначала простейший случай размещения нулей во входном слове: $e_1 \oplus R_2(e_1)$, $e_1 \oplus R_4(e_1)$, $e_1 \oplus R_8(e_1)$ и так далее, а также $R_1(e_1 \oplus R_2(e_1))$, $R_1(e_1 \oplus R_4(e_1))$, $R_1(e_1 \oplus R_8(e_1))$. Для таких входных слов один уровень вычислительной

Таблица 7

Выход на уровне	ПС	ППС	ПС	ППС
	Входной код 0001 1111		*Входной код* 0000 0111	
1	1001 0000	1110 1110	1000 0100	0111 0111
2	1101 1000	0101 0101	1100 0110	1010 1010
3	1011 0100	1111 1111	1010 0101	1111 1111
4	1110 1110		0111 0111	
5	1001 1001		1100 1100	
6	0101 0101		1010 1010	
7	1111 1111		1111 1111	
	Входной код 0011 1110		*Входной код* 1100 0001	
1	0010 0001	1011 0001	0010 0001	1011 0001
2	1011 0001	0101 0101	1011 0001	1010 1010
3	0110 1001	1111 1111	0110 1001	1111 1111
4	1101 1101		1101 1101	
5	0011 0011		0011 0011	
6	1010 1010		1010 1010	
7	1111 1111		1111 1111	

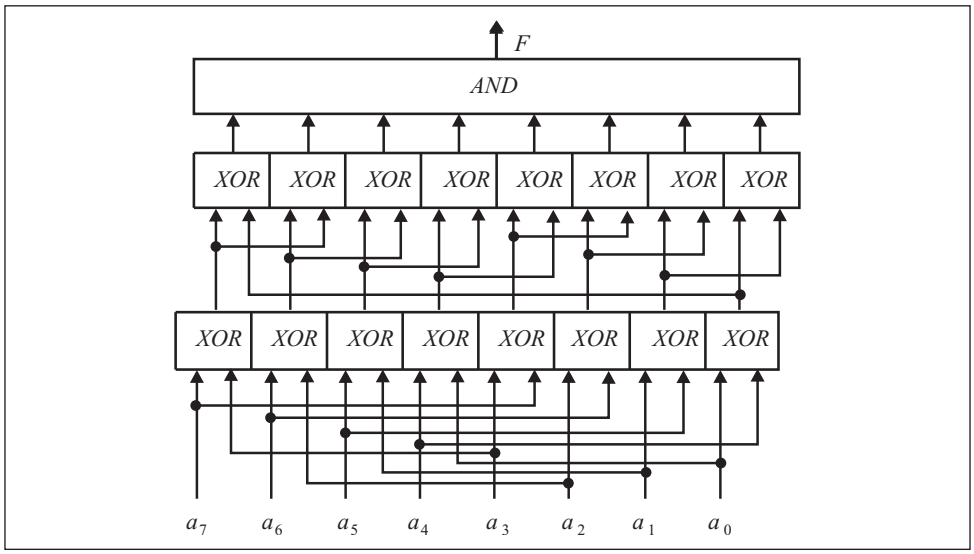

Рис. 5. Схема ППС преобразователя на основе оператора сдвига R_4

структуры уже реализован: для первого слова — R_2, для второго — R_4, для третьего — R_8 и так далее.

Теперь рассмотрим общий случай расположения нулей во входном слове. Пусть это i-й и j-й символы, где $i< j$. Тогда возможны два случая:

1) $j = i + 2k$, т.е. $(j - i = 2k)$ — четное;

2) $j = i + 2k + 1$, т.е. $(j - i = 2k + 1)$ — нечетное.

В случае 1 задача сводится к редукции слова $e_1 \oplus R_{2k}(e_1)$, поскольку $\bar{J} = R_{i-1}(e_1) \oplus R_{i-1+2k}(e_1) = R_{i-1}(e_1 \oplus R_{2k}(e_1))$. Этот случай аналогичен рассмотренным в теореме 6. Действительно,

если $k = 1$, то $\bar{J} = e_1 \oplus R_2(e_1)$;

если $k = 2$, то $\bar{J} = e_1 \oplus R_4(e_1)$;

если $k = 3$, то $\bar{J} = e_1 \oplus R_6(e_1)$;

если $k = 4$, то $\bar{J} = e_1 \oplus R_8(e_1)$ и так далее.

Отсюда также следует, что если i — четное, то $R_i(\bar{J}) = V$, иначе — $R_i(\bar{J}) = V_1$.

В случае 2 задача сводится к редукции слова $e_1 \oplus R_{2k+1}(e_1) = J$. Преобразуем это слово с помощью оператора R_1 к слову $e_1 \oplus R_1(e_1) \oplus \oplus R_{2k+1}(e_1) \oplus R_{2k+2}(e_1)$. Применив к этому слову оператор R_4, получим

$$e_1 \oplus R_1(e_1) \oplus R_{2k+1}(e_1) \oplus R_{2k+2}(e_1) \oplus$$

$$\oplus R_4(e_1) \oplus R_5(e_1) \oplus R_{2k+5}(e_1) \oplus R_{2k+6}(e_1).$$

Если $n=8,\ k=1,$ то данное слово равно V_1, т.е. если i — четное, а k — нечетное, то на выходе вычислительной структуры будет слово V_1.

Если $n=8,\ k=2,$ то $J=e_1\oplus R_5(e_1)$ и на выходе вычислительной структуры получим $e_1\oplus R_2(e_1)\oplus R_4(e_1)\oplus R_6(e_1)=V$, т.е., если i и k — четные, то на выходе вычислительной структуры получаем слово V.

Обобщает изложенное следующая теорема.

Теорема 7. Если во входном слове нули находятся на i-м и j-м местах, то на выходе получаем следующее:

1) V при условии четных i и j, иначе — V_1;

2) V_1 при нечетных $(i-j=2k+1)$, иначе — V.

В случае 1 вычислительная структура должна иметь $\log_2 n-2$ уровней (см. табл. 8, входные коды 0101 1111 и 1010 1111). Параллельно-последовательная структура для их реализации показана на рис. 5. В случае 2 вычислительная структура должна иметь $\log_2 n-1$ уровней (см. табл. 8, входные коды 0110 1111 и 0111 0111), а ППС для их реализации представлена на рис. 4, *б*.

Доказательство теоремы 7 следует из рассмотренных выше случаев и табл. 8.

Если вычислительная структура использует не все уровни (при любом варианте реализации — ПС или ППС), то возможны два подхода.

1. Незадействованные уровни настраиваются на реализацию логической функции A. Общее число уровней остается неизменным, что приводит к дополнительным временным задержкам и аппаратным затратам.

Таблица 8

Выход на уровне	ПС	ППС	ПС	ППС
	Входной код 0101 1111		*Входной код* 1010 1111	
1	1111 0000	1010 1010	0111 1000	0101 0101
2	1000 1000	1111 1111	0100 0100	1111 1111
3	1100 1100		0110 0110	
4	1010 1010		0101 0101	
5	1111 1111		1111 1111	
	Входной код 0110 1111		*Входной код* 0111 0111	
1	1101 1000	1101 1000	0100 0001	0100 0001
2	1011 0100	0101 0101	1110 0001	0101 0101
3	1110 1110	1111 1111	0001 0001	1111 1111
4	1001 1001		1001 1001	
5	0101 0101		0101 0101	
6	1111 1111		1111 1111	

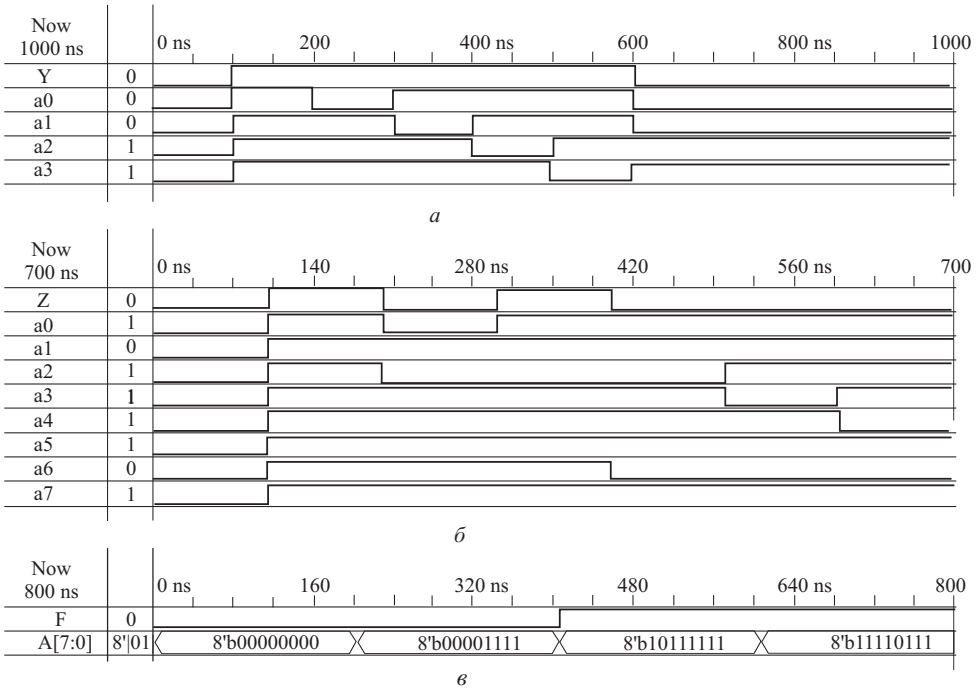

Рис. 6. Временные диаграммы работы ПС (*а*), ППС (*б*) на основе операции *AND* и ППС на основе операции *XOR* (*в*)

2. Незадействованные уровни структуры исключаются при реализации (например, структура, представленная на рис. 5). Число уровней сокращается, что приводит к уменьшению временных задержек и аппаратных затрат.

Структуры на основе кристаллов ПЛИС. Реализация ПС и ППС преобразователя циклических кодов на основе операций *AND* и *XOR* с использованием кристалла серии Virtex-4 (XC4VLX15) фирмы XILINX выполнена с помощью инструментального пакета САПР ПЛИС (ISE Foundation) [4]. Временные диаграммы, полученные в результате моделирования этих структур и подтверждающие правильность их функционирования, приведены на рис. 6.

Выводы

Для преобразования циклических кодов с расстоянием Хемминга 1 относительно эталонного слова, состоящего из единиц, при реализации ПС на основе операции *AND* требуется $r_1 = (m-1)$ уровней, а при реализации

ППС — $r_2 = \log_2 n$ уровней. Таким образом, выигрыш в аппаратных затратах при реализации структур преобразователей циклических кодов, а также времени преобразования определяется относительной величиной $\Delta r = r_1 / r_2 = (m-1)/(\log_2 n)$.

Для циклических кодовых слов, содержащих два и более нулей, реализация ПС на основе логической функции *AND* при преобразовании циклических кодов требует $r_1 = (d_1 - 1)$ уровней, остальные уровни структуры настраиваются на реализацию логической функции *A*; реализация ППС требует $r_2 = \log_2 n - 1$ уровней.

Для реализации ППС преобразователя на основе операции *XOR* циклических кодов в зависимости от числа и местоположения нулей во входном слове требуется в общем случае $\log_2 n - 2$ или $\log_2 n - 1$ уровней, что следует из теорем 5—7.

Таким образом, выигрыш в аппаратных затратах и времени при реализации ППС преобразователей циклических кодов на основе операции *XOR* по сравнению с ПС преобразователями имеет логарифмическую зависимость.

Рассмотренные структуры могут быть использованы для решения задач помехоустойчивого кодирования и распознавания образов, где в качестве меры близости используется расстояние Хемминга.

The correctness of functioning of multilevel structure (of sequential and parallel-sequential types) of converter of cyclic code words containing the cyclic group of contiguous units on the basis of Boolean functions of *AND* and *XOR* is substantiated. The offered structures are implemented in the PLIC element basis.

СПИСОК ЛИТЕРАТУРЫ

1. *Питерсон У., Уэлдон Э.* Коды, исправляющие ошибки: Пер. с англ. — М. : Мир, 1976. — 594 с.
2. *Брейтон Р.К., Хэтчел Г.Д., Санджованни А.Л.* Синтез многоуровневых комбинационных логических схем. — Винчентелли // ТИИЭР. — 1990. — **78**, № 2. — С. 38—83.
3. *Opanasenko V.N., Kryvyi S.L.* Partitioning the Full Range of Boolean Functions Based on the Threshold and Threshold Relation // Cybernetics and Systems Analysis. —2012. — Vol. 48, № 3. — P. 459—468.
4. *Palagin A.V., Opanasenko V.N.* Design and Application of the PLD-based Reconfigurable Devices // Design of Digital Systems and Devices. — Springer, Verlag, Berlin, Heidelberg. — 2011. — Vol. 79. — P. 59—91.

Поступила 03.02.14

ПАЛАГИН Александр Васильевич, академик НАН Украины, зам. директора по науке Ин-та кибернетики им. В.М. Глушкова НАН Украины. В 1961 г. окончил Киевский политехнический ин-т. Область научных исследований — теория универсальных и специализированных компьютерных систем; научно-методические основы построения интеллектуальных систем информационно-когнитивной поддержки научных исследований, интеллектуальных сетей и систем массового информационного сервиса; компьютерные системы с виртуальной архитектурой; технологии системной интеграции в задачах исследовательского проектирования.

ОПАНАСЕНКО Владимир Николаевич, д-р техн. наук, вед. науч. сотр., профессор Ин-та кибернетики им. В.М. Глушкова НАН Украины. В 1979 г. окончил Казанский авиационный ин-т. Область научных исследований — архитектурно-структурная организация реконфигурируемых устройств и компьютерных систем с реконфигурируемой архитектурой.

КРЫВЫЙ Сергей Лукьянович, д-р физ.-мат. наук, профессор Киевского национального университета им. Т.Г. Шевченко, который окончил в 1972 г. Область научных исследований — дискретная математика, теория автоматов и сетей Петри, верификация программ, теория сложности алгоритмов, нейронные сети.