УДК 004.052.32 + 681.518.5 DOI: 10.17587/it.25.426-434

Д. В. Ефанов, д-р техн. наук, доц., e-mail: TrES-4b@yandex.ru,
ООО "ЛокоТех-Сигнал",
Российский университет транспорта (РУТ (МИИТ)), Москва

Троичный код паритета в системах рабочего диагностирования устройств автоматики и вычислительной техники

В статье обозначен интерес к развитию методов рабочего диагностирования устройств автоматики, функционирующих в троичной логике. Представлены аналогии между подходами к организации систем рабочего диагностирования устройств двоичной и троичной логики. Описан троичный код паритета и некоторые его свойства, учет которых целесообразен при организации систем диагностирования. Доказано, что все информационные векторы кода паритета распределены равномерно между всеми его контрольными векторами. Приведена формула подсчета числа необнаруживаемых ошибок в информационных векторах троичных кодов с равномерным распределением всех информационных векторов между всеми контрольными векторами. Доказана теорема о коде с наименьшим общим числом необнаруживаемых ошибок в информационных векторах троичных кодов для фиксированных значений длин информационных и контрольных векторов. Приведены правила построения модифицированных троичных кодов паритета.

Ключевые слова: *автоматика и вычислительная техника, троичная логика, рабочее диагностирование, результаты вычисления, косвенный контроль, ошибки на выходах, троичный код паритета*

Введение

Устройства автоматики и вычислительной техники в современном мире реализуются на основе компонентов, использующих принципы двоичного представления сигналов. Высокий уровень потенциала соответствует сигналу "логическая 1", а низкий — сигналу "логический 0" (или наоборот). Широкое распространение элементы, реализующие принципы двоичной логики, получили благодаря простоте реализации и вполне понятным принципам обеспечения надежности функционирования. Более того, признание основной именно двоичной логики для реализации вычислительных систем было зафиксировано в известной архитектуре фон Неймана [1].

Тем не менее, за годы развития компьютерной техники во второй половине XX — первой четверти XXI века были удачные попытки реализации вычислительных машин, функционирующих на принципах троичной логики [2]. Примерами тому являются троичные вычислительные системы "Сетунь", реализованная в МГУ в 1958 г. (под руководством Н. П. Брусенцова) [3, 4], и TCA2 (версия v2.0), созданная в Калифорнийском политехническом университете в 2008 г. (Дж. Коннели, К. Пател, А. Чавез) [5]. В ряде публикаций [6—8] используется идея реализации устройств троичной логики, "собранных" на традиционных устройствах двоичной логики. О преимуществе троичной логики перед двоичной говорят некоторые разработчики квантовых компьютеров, предлагая использовать вместо кубитов информации кутриты, объясняя это серьезным уменьшением числа квантовых вентилей [9]. Кроме того, троичная логика может оказаться эффективной для информационной защиты современных устройств, использующих IoT [10].

Преимущества троичной логики перед двоичной (и перед любой другой) объясняются более плотной записью чисел. Кроме того, троичная логика "покрывает" двоичную и может использовать все ее преимущества, а вычислительные устройства, реализованные на троичной логике, должны оказаться более быстродействующими [11].

Развитие компьютерных технологий и разработка киберфизических систем управле-

ния на фоне колоссального прогресса в области информатизации [12] подогревают интерес большого числа исследователей цифровых устройств, реализуемых в троичной логике [13—19]. Параллельно развитию самой троичной техники и методов ее описания должны развиваться методы технической диагностики систем, реализованных в троичной логике [20, 21].

В данной работе проводится параллель между элементарными подходами в технической диагностике (с упором на методы рабочего диагностирования [22]) традиционных устройств, функционирующих в двоичной (бинарной) логике, и устройств, реализованных в троичной (тернарной) логике (*ternary logic* или *three-valued logic*).

1. Ошибки на выходах логических устройств

Будем рассматривать цифровые устройства, реализованные в троичной логике. При этом следует пояснить тот факт, что в троичной логике существуют несколько вариантов систем обозначения и трактовки логических сигналов: (0, 1, 2), (−1, 0, 1), (0, 1/2, 1) и пр. В общем смысле троичная система счисления реализуется в двух вариантах, образуя несимметричную и симметричную системы счисления. В несимметричной системе счисления используются обозначения логических сигналов (0, 1, 2). В симметричной системе счисления знаки обозначения логических сигналов следующие: (−1, 0, 1). Вообще, символы, обозначающие логические уровни для устройств автоматики троичной логики, могут быть использованы любые (с оговоркой старшинства). Далее в рассуждениях будем использовать следующий набор символов: $x, f \in \{i; 0; 1\}$, ориентируясь на вариант симметричной системы счисления. Следует отметить, что все описанные далее результаты универсальны и не привязаны к конкретному виду системы обозначения, а сам символ третьего сигнала в виде "*i*" использован для простоты записи троичных векторов.

Один трит информации может принимать значения из множества $f \in \{i; 0; 1\}$, другими словами, иметь три варианта значений. На рис. 1 изображено некоторое устройство, реализованное на элементах троичной логики (они обозначены на рисунке как *TG — ternary gate*), на выходах которого реализуется троичный вектор $<f_1\ f_2\ \ldots\ f_{m-1}\ f_m>$ (общее число таких троичных векторов определяется величиной 3^m). В процессе функционирования

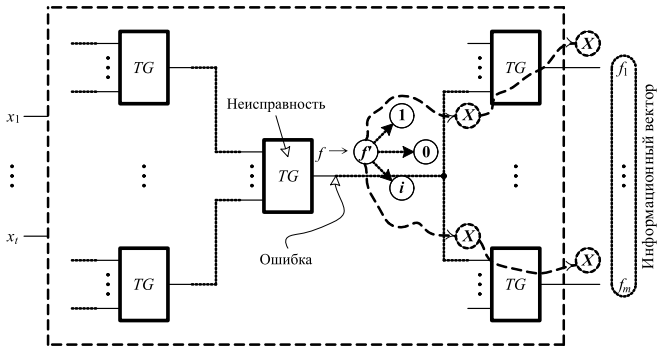

Рис. 1. Распространение ошибок на выходы устройства

устройства не исключены физические дефекты (неисправности), природа которых может быть самой разнообразной (например, в виду электромагнитных помех). Неисправности приводят к установлению на линиях схемы устройства ложных сигналов, а это, в свою очередь, является причиной трансляции неверных результатов вычислений на рабочие выходы устройства (на рис. 1 неизвестный сигнал показан символом "X" — это может быть как верный сигнал при компенсации ошибки, так и неверный сигнал). Ошибка в вычислениях на какой-либо линии схемы устройства приводит к формированию на его выходах ошибочного информационного вектора.

Элементарной моделью неисправностей для устройств, реализованных в троичной логике, по аналогии с двоичной логикой может служить модель одиночной константной неисправности (*stuck-at fault*). Суть ее сводится к тому, что в любой момент времени в устройстве возможна только одиночная неисправность, приводящая к установлению на выходах какого-либо элемента ложного сигнала *i*, 0 или 1. На рис. 2 в качестве примера представлены возможные варианты искажений значений на выходах логических элементов троичной и двоичной логики — соответственно трита и бита информации.

Таким образом, внутренние ошибки устройства приводят к искажениям значений выходного информационного вектора. Общее число таких ошибок определяется величиной $N_m^3 = 3^m(3^m - 1)$, где m — длина информационного вектора. Для сравнения число вариантов искажений двоичного информационного вектора длиной m определяется по формуле $N_m^2 = 2^m(2^m - 1)$. К примеру, при $m = 3$ для двоичного вектора имеем $N_3^2 = 56$ вариантов искажений, а для троичного — $N_3^3 = 702$ (более чем в 12 раз!). С увеличением значения m число вариантов искажений троичных информационных векторов существенно растет (не-

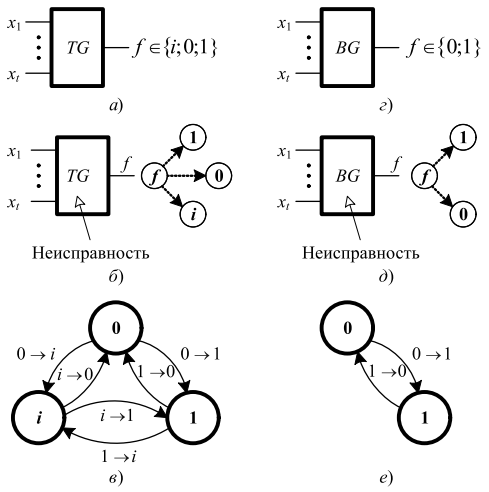

Рис. 2. Ошибки на выходах логических элементов троичной и двоичной логики (ошибки в тритах и битах):

а — троичный логический элемент (*TG — ternary gate*); *б* — искажения функций троичного логического элемента; *в* — граф искажений значений трита; *г* — двоичный логический элемент (*BG — binary gate*); *д* — искажения функций двоичного логического элемента; *е* — граф искажений значений бита

сравнимо с числом вариантов искажений двоичных векторов):

$$\lim_{m\to\infty}\chi_m = \lim_{m\to\infty}\frac{N_m^2}{N_m^3} = \lim_{m\to\infty}\frac{2^m(2^m-1)}{3^m(3^m-1)} = \\ = \lim_{m\to\infty}\left(\frac{2}{3}\right)^m \lim_{m\to\infty}\frac{2^m-1}{3^m-1} = 0. \qquad (1)$$

Уже при $m = 5$ значение $\chi_5 = 0{,}01687$, а при $m = 10$ $\chi_{10} = 0{,}0003$.

Более того, ошибки в троичных информационных векторах гораздо "богаче" ошибок в двоичных информационных векторах (это как раз следует из многообразия вариантов искажений одного трита).

Возникает следующая задача: определить наличие неисправности в устройстве, рассматривая его как "черный ящик" и анализируя только выходные его значения. Эта задача решается методами рабочего диагностирования [23, 24]. В двоичной логике широко распространено применение избыточного кодирования для решения задачи рабочего диагностирования [25—28]. Избыточное кодирование может быть применено и для решения аналогичной задачи в троичной логике.

2. Троичный код паритета

Избыточный код, ориентированный на обнаружение ошибок в троичных информационных векторах, может быть построен путем использования элементарных троичных функций. В табл. 1 представлены описания основных используемых функций троичной логики (всего в троичной логике функций от двух переменных может быть построено $3^{3^2} = 3^9 = 19\,683$). К слову, теория построения троичных кодов

Таблица 1

Описание троичных функций от двух переменных

Название функций	Обозначение	x_1x_2								
		ii	*i*0	*i*1	0*i*	00	01	1*i*	10	11
Конъюнкция	$x_1 \cdot x_2$	*i*	*i*	*i*	*i*	0	0	*i*	0	1
Дизъюнкция	$x_1 \vee x_2$	*i*	0	1	0	0	1	1	1	1
Логическое умножение по модулю три	$x_1 \otimes x_2$	1	0	*i*	0	0	0	*i*	0	1
Логическое сложение по модулю три	$x_1 \oplus x_2$	1	*i*	0	*i*	0	1	0	1	*i*
Функция Вебба	$x_1 \mid x_2$	0	1	*i*	1	1	*i*	*i*	*i*	*i*
Пороговое сложение	$x_1 + x_2$	*i*	*i*	0	*i*	0	1	0	1	1
Исключающий максимум	$x_1 \Uparrow x_2$	*i*	0	1	0	*i*	1	1	1	*i*
Среднее	$x_1 \to x_2$	*i*	0	*i*	0	1	0	*i*	0	*i*
Сравнение	$x_1 \mathrm{cmp} x_2$	0	*i*	*i*	1	0	*i*	1	1	0
Сильная конъюнкция	$x_1 \,\&_{\mathrm{L}}\, x_2$	*i*	*i*	*i*	*i*	*i*	0	*i*	0	1
Импликация Лукасевича	$x_1 \to_{\mathrm{L}} x_2$	1	1	1	0	1	1	*i*	0	1
Конъюнкция Клини	$x_1 \wedge_{+} x_2$	*i*	0	*i*	0	0	0	*i*	0	1
Импликация Клини	$x_1 \to_{+} x_2$	1	0	*i*	1	0	0	1	0	1
Импликация Гейтинга (Геделя)	$x_1 \to_{\mathrm{G}} x_2$	1	1	1	*i*	1	1	*i*	0	1
Материальная импликация	$x_1 \to_{\mathrm{M}} x_2$	1	1	1	0	0	1	*i*	0	1
Функция следования Брусенцова	$x_1 \to_{\mathrm{B}} x_2$	1	0	0	0	0	0	*i*	0	1
Тождество	$x_1 \equiv x_2$	1	*i*	*i*	*i*	1	*i*	*i*	*i*	1

активно развивается [29—32], известны и методы построения блочных равномерных троичных кодов, например, аналога равновесного кода из бинарной логики, так называемого композиционного троичного кода [33, 34].

Для решения задач технической диагностики, в том числе рабочего диагностирования в условиях необходимости снижения избыточности устройства и упрощения технических средств диагностирования, полезными могут оказаться простейшие избыточные коды. Наипростейшим является *троичный код паритета (ternary parity code)*. Он строится по аналогии с двоичным кодом паритета путем использования функции *свертки по модулю три*, или *логического сложения по модулю три*:

$$g = f_1 \oplus f_2 \oplus \ldots \oplus f_{m-1} \oplus f_m. \qquad (2)$$

Таким образом, троичный код паритета имеет всего один контрольный бит, что позволяет обнаруживать любые одиночные ошибки в информационных векторах.

Известны практические реализации функции сложения по модулю три на традиционных микроэлектронных компонентах. Например, на рис. 3 приведена принципиальная схема реализации функции сложения по модулю три из работы [35], реализованная на элементах двоичной логики. В схеме присутствуют транзисторы VT_1 и VT_2, представляющие собой пару взаимно дополнительных усилителей. Сигнал на выходе этой пары совпадает по фазе с входным сигналом и вдвое больше его по амплитуде. Напряжение смещения транзистора VT_3 выбирается таким образом, чтобы сигнал через сопротивления R_1 и R_2 переключал ток, протекающий через элементы VD_1 и R_3, что приводит к увеличению выходного тока транзистора VT_2.

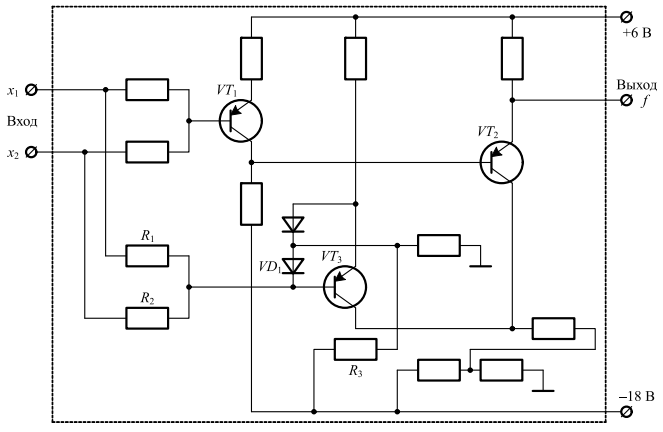

Рис. 3. Принципиальная схема устройства сложения по модулю три

Таблица 2

Описание функции сложения по модулю три

x_1	x_2		
	i	0	1
i	1	i	0
0	i	0	1
1	0	1	i

Ключевым отличием функции сложения по модулю три от остальных часто используемых функций (табл. 1) является тот факт, что ровно на трети входных наборов она принимает значения i, на трети входных наборов — значения 0 и на трети входных наборов — значения 1 (другими словами, распределение значений равномерно относительно всех входных наборов). Компактное описание функции сложения по модулю три приведено в табл. 2.

Обозначим троичный код паритета как $P^T(m, k)$-код, где k — число контрольных разрядов. Пример такого кода ($P^T(3, 1)$-кода) приведен в табл. 3.

Таблица 3

Кодовые векторы $P^T(3,1)$-кода

№	f_1	f_2	f_3	g
1	i	i	i	0
2	i	i	0	1
3	i	i	1	i
4	i	0	i	1
5	i	0	0	i
6	i	0	1	0
7	i	1	i	i
8	i	1	0	0
9	i	1	1	1
10	0	i	i	1
11	0	i	0	i
12	0	i	1	0
13	0	0	i	i
14	0	0	0	0
15	0	0	1	1
16	0	1	i	0
17	0	1	0	1
18	0	1	1	i
19	1	i	i	i
20	1	i	0	0
21	1	i	1	1
22	1	0	i	0
23	1	0	0	1
24	1	0	1	i
25	1	1	i	1
26	1	1	0	i
27	1	1	1	0

Следует отметить важную закономерность, присущую $P^T(m, k)$-кодам.

Теорема 1. *$P^T(m, k)$-коды имеют равномерное распределение информационных векторов между всеми контрольными векторами.*

Доказательство справедливости положения теоремы 1 тривиально и вытекает из того факта, что функция сложения по модулю три, используемая при вычислении контрольного разряда, имеет равномерное распределение относительно всех входных наборов. При построении кода как раз и рассматриваются все входные векторы. *Теорема доказана.*

Положение теоремы, несмотря на его простоту, в теории рабочего диагностирования логических устройств играет важную роль. Код, обладающий свойством равномерности распределения информационных векторов между всеми возможными контрольными векторами, будет иметь наименьшее число необнаруживаемых в информационных векторах ошибок среди всех возможных кодов для данных значений m и k.

Теорема 2. *Троичный код с параметрами m и k будет обладать минимальным общим числом необнаруживаемых ошибок при условии, что все 3^m информационных вектора будут равномерно распределены между всеми 3^k контрольными векторами, а общее число необнаруживаемых ошибок в таком коде будет определяться по формуле*

$$N_{m,k}^{\min} = 3^m\left(3^{m-k} - 1\right). \qquad (3)$$

Доказательство. Число различных контрольных векторов, или *контрольных групп*, определяется числом разрядов в контрольных векторах и равно 3^k. Так как всего существуют 3^m информационных вектора, в каждой контрольной группе будет размещено ровно по $q = \dfrac{3^m}{3^k} = 3^{m-k}$ информационных вектора. Число необнаруживаемых ошибок в одной такой контрольной группе будет равно $q(q-1) = 3^{m-k}(3^{m-k} - 1)$. Так как число контрольных групп равно 3^k, то умножая на данную величину полученное ранее выражение, получаем выражение (3).

Докажем, что именно формула (3) определяет минимальное общее число необнаруживаемых в информационных векторах ошибок. Для этого покажем, что минимальное нарушение равномерности распределения информационных векторов между контрольными векторами приведет к увеличению числа необнаруживаемых ошибок по сравнению с рассчитанным по формуле (3).

Пусть в $3^k - 2$ контрольных группах размещено ровно по $q = 3^{m-k}$ информационных векторов, а в двух оставшихся размещены $q - 1 = 3^{m-k} - 1$ и $q + 1 = 3^{m-k} + 1$ векторов соответственно. Не будем рассматривать $3^k - 2$ контрольные группы с $q = 3^{m-k}$ информационными векторами, а остановимся на подсчете общего числа необнаруживаемых ошибок в двух оставшихся группах. Суммарно это следующее число необнаруживаемых ошибок:

$$\begin{aligned} Q_1 &= (3^{m-k} - 1)(3^{m-k} - 1 - 1) + \\ &+ (3^{m-k} + 1)(3^{m-k} + 1 - 1) = \\ &= (3^{m-k} - 1)(3^{m-k} - 2) + (3^{m-k} + 1)3^{m-k} = \\ &= 3^{2(m-k)} - 3 \cdot 3^{(m-k)} + 2 + 3^{2(m-k)} + 3^{(m-k)} = \\ &= 2 \cdot 3^{2(m-k)} - 2 \cdot 3^{(m-k)} + 2. \end{aligned} \qquad (4)$$

Для двух групп с $q = 3^{m-k}$ информационными векторами число необнаруживаемых ошибок равно

$$Q_2 = 2 \cdot 3^{m-k}\left(3^{m-k} - 1\right) = 2 \cdot 3^{2(m-k)} - 2 \cdot 3^{m-k}. \qquad (5)$$

Найдем значение величины $\Delta = Q_1 - Q_2$:

$$\begin{aligned} \Delta &= \left(2 \cdot 3^{2(m-k)} - 2 \cdot 3^{(m-k)} + 2\right) - \\ &- \left(2 \cdot 3^{2(m-k)} - 2 \cdot 3^{m-k}\right) = 2. \end{aligned} \qquad (6)$$

Значение величины $\Delta = 2 > 0$ при любых значениях m и k. Другими словами, число необнаруживаемых ошибок в результате нарушения равномерности увеличилось на 2. Еще большее нарушение равномерности распределения всех 3^m информационных векторов между всеми 3^k контрольными векторами приводит только к увеличению числа необнаруживаемых ошибок.

Таким образом, минимальное общее число необнаруживаемых ошибок в троичном коде с параметрами m и k определяется формулой (3). Это, в свою очередь, означает, что код с минимальным общим числом ошибок в информационных векторах будет иметь равномерное распределение информационных векторов между всеми контрольными векторами, *что и требовалось доказать.*

В табл. 4 все информационные векторы $<f_1f_2f_3>$ $P^T(3,1)$-кода распределены между всеми однотритными контрольными векторами. Распределение равномерно.

Таблица 4

Распределение информационных векторов на контрольные группы для $P^T(3,1)$-кода

g		
i	0	1
$<f_1f_2f_3>$		
$ii1z$	iii	$ii0$
$i00$	$i01$	$i0i$
$i1i$	$i10$	$i11$
$0i0$	$0i1$	$0ii$
$00i$	000	001
011	$01i$	010
$1ii$	$1i0$	$1i1$
101	$10i$	100
110	111	$11i$

Анализ контрольных групп, соответствующих контрольным векторам кода, позволяет подсчитать общее число необнаруживаемых кодом ошибок. В каждой контрольной группе такого кода находится по $\frac{3^m}{3} = 3^{m-1}$ информационных вектора. Искажение не будет обнаружено кодом в том случае, если информационный вектор одной группы перейдет в результате искажения в информационный вектор другой группы. Число таких переходов внутри группы равно $3^{m-1}(3^{m-1}-1)$. Так как группы три, то общее число необнаруживаемых рассматриваемым кодом ошибок равно $N^3_{m,1} = 3 \cdot 3^{m-1}\left(3^{m-1}-1\right) = 3^m\left(3^{m-1}-1\right)$. Троичным кодом паритета не обнаруживаются $N^3_{3,1} = 3^3\left(3^{3-1}-1\right) = 27 \cdot 8 = 216$ ошибок в информационных векторах (среди них 162 двукратных и 54 трехкратных ошибок). Для сравнения, бинарный аналог данного кода — код паритета — не обнаруживает $N^2_{m,1} = 2^m\left(2^{m-1}-1\right)$ ошибок в информационных векторах. Для трехбитного вектора это число равно $N^2_{3,1} = 24$ ошибки, что в девять раз меньше, чем в троичном коде паритета для той же длины информационного вектора.

Определим, какую долю ошибок в информационных векторах можно обнаружить с помощью троичного кода паритета:

$$\lim_{m\to\infty}\gamma_{m,k} = \lim_{m\to\infty}\frac{N^3_{m,1}}{N^3_m} = \\ = \lim_{m\to\infty}\frac{3^m(3^{m-1}-1)}{3^m(3^m-1)} = \lim_{m\to\infty}\frac{3^{m-1}-1}{3^m-1} = \frac{1}{3}. \quad (7)$$

Бинарный код паритета не обнаруживает в пределе $m \to \infty$ 50 % ошибок, тогда как тернарный код паритета — 33,3 %.

Полученный результат показывает аналогию между кодами паритета в двоичной и троичной логике, а также говорит о том, что доля необнаруживаемых $P^T(m, k)$-кодами ошибок от общего их числа в информационных векторах меньше, чем аналогичный показатель для двоичного кода паритета. Тем не менее, троичные коды паритета могут быть использованы при разработке технических средств диагностирования логических устройств с учетом основной своей особенности — обнаружения однократных искажений. Это позволяет транслировать известные подходы по применению двоичных кодов паритета [36, 37] при организации систем рабочего диагностирования. При этом следует установить особенности возникающих в троичных информационных векторах (и на рабочих выходах объектов диагностирования) ошибок, подобно тому, как это сделано в работе [38] для бинарных разделимых кодов.

Для построения $P^T(3,1)$-кода используется функция $g = f_1 \oplus f_2 \oplus f_3$. При синтезе кодера такого кода потребуется каскадное подключение двух сумматоров по модулю три (рис. 3). Для улучшения характеристик обнаружения ошибок в информационных векторах без внесения дополнительной сложности в кодер может быть использовано выделение двух контрольных тритов: $g_1 = f_1 \oplus f_2$ и $g_2 = f_2 \oplus f_3$. Для построенного таким образом $P^T(3,2)$-кода распределение информационных векторов между всеми контрольными векторами представлено в табл. 5. Согласно формуле (3) $P^T(3,2)$-кодом не будет обнаруживаться 54 ошибки, что в четыре раза меньше, чем число ошибок, обнаруживаемых $P^T(3,1)$-кодом. Кроме того, в отличие от $P^T(3,1)$-кода у $P^T(3,2)$-кода все векторы внутри контрольных групп имеют кодовое расстояние Хэмминга $d = 3$, а значит, ими

Таблица 5

Распределение информационных векторов на контрольные группы для $P^T(3,2)$-кода

$<g_1g_2>$								
ii	$i0$	$i1$	$0i$	00	01	$1i$	10	11
$<f_1f_2f_3>$								
$i0i$	$i00$	101	$i11$	$i1i$	$i10$	$ii0$	$ii1$	iii
$0i0$	$0i1$	$0ii$	$00i$	000	001	011	$01i$	010
111	$11i$	110	$1i0$	$1i1$	$1ii$	$10i$	100	101

не обнаруживаются только трехкратные ошибки в информационных векторах (всего 54 трехкратных ошибки). Таким образом, распределение необнаруживаемых ошибок по кратностям, что немаловажно, смещено в сторону ошибок большей кратности.

Отметим, что способ выделения двух контрольных трит можно использовать аналогично контролю двух групп выходов устройства. Для кодов же с большим числом информационных разрядов можно строить модифицированные коды паритета с тремя и более контрольными разрядами (в том числе аналоги двоичных полиномиальных кодов и кодов Рида—Маллера [39]).

В заключение следует обратить внимание и на то, что представленные правила не являются единственными и возможны другие альтернативные способы построения кодов паритета. Например, в работе [40] отмечается, что известен способ построения кода с выделением двух контрольных битов, каждый из которых предназначен для отдельной проверки четности суммы чисел "*i*" и "1".

Заключение

При синтезе технических средств диагностирования для устройств автоматики, функционирующих в троичной логике, могут эффективно применяться троичные коды, ориентированные на обнаружение ошибок в информационных векторах. Одним из эффективных кодов для этих целей является троичный код паритета. Его можно применять при реализации технических средств рабочего диагностирования следующими способами. Первый состоит в том, что осуществляется поиск групп независимых выходов устройств и строятся отдельные схемы контроля результатов вычислений, а контрольные выходы получаемых схем контроля объединяются на входах самопроверяемых компараторов троичных сигналов. Второй способ связан с преобразованиями структур объектов диагностирования в структуры с независимыми выходами и контролем общей группы на основе кода паритета.

Троичный код паритета относится к троичным кодам с наименьшим общим числом необнаруживаемых ошибок в информационных векторах для своих параметров m и k. Могут быть также построены такие коды с улучшенными характеристиками обнаружения ошибок в информационных векторах на основе выбора нескольких функций свертки по модулю три и образования из них контрольных векторов.

Использование модифицированных троичных кодов паритета может оказаться эффективным с позиции снижения затрат на реализацию технических средств диагностирования по сравнению с дублированием.

Список литературы

1. **Aspray W.** John von Neumann and the Origins of Modern Computing (History of Computing). Boston, Cambridge: MIT, 1990, 396 p.

2. **Карпенко А. С.** Развитие многозначной логики. М.: Издательство ЛКИ, 2010, 448 с.

3. **Брусенцов Н. П.** Отчего математическая логика несодержательна // Историко-математические исследования / Под редакцией С. С. Демидова. Вторая серия. Вып. 11 (46), М.: Янус-К, 2006. С. 228—234.

4. **Брусенцов Н. П., Маслов С. П., Розин В. П., Тишулина А. М.** Малая цифровая вычислительная машина "Сетунь". М.: Издательство МГУ, 1962. 140 с.

5. **Connely J.** Ternary Computing Testbed 3-Trit Computer Architecture. California Polytechnic State University of San Luis Obispo, August 29th, 2008. 184 p.

6. **Roy D., Merril Jr.** Ternary Logic in Digital Computers // Proceedings of the SHARE design automation project (DAC '65), ACM New York, NY, USA. P. 6.1—6.17. DOI: 10.1145/800266.810759.

7. **Hu M., Smith K. C.** Self-Checking Binary Logic Systems Using Ternary Logic Circuits // Canadian Electrical Engineering Journal. 1984. Vol. 9, Iss. 3. P. 100—104. DOI: 10.1109/CEEJ.1984.6593793.

8. **Wu J.** Ternary Logic Circuit for Error Detection and Error Correction // Proceedings of 19th International Symposium on Multiple-Valued Logic. 29—31 May 1989. Guangzhou, China. P. 94—99. DOI: 10.1109/ISMVL.1989.37766.

9. **Lanyon B. P., Barbieri M., Almeida M. P., Jennewein T., Ralph T. C., Resch K. J., Pryde G. J., O'Brien J. L., Gilchrist A., White A. G.** Simplifying Quantum Logic Using Higher-Dimensional Hilbert Spaces // Nature Physics. 2009. Vol. 5, Iss. 2. P. 134—140. DOI: 10.1038/nphys1150.

10. **Cambou B., Flikkema P. G., Palmer J., Telesca D., Philabaum C.** Can Ternary Computing Improve Information Assurance? // Cryptography. 2018. Vol. 2, Iss. 1 (March 2018). P. 1—16. DOI: 10.3390/cryptography2010006.

11. **Петров А.** Троичный компьютер: да, нет, может быть // Популярная механика. 2011. № 9. С. 72—76.

12. **Hahanov V.** Cyber Physical Computing for IoT-driven Services. New York, Springer International Publishing AG, 2018. 279 p.

13. **Vudadha C., Katragadda S., Phaneendra P. S.** 2:1 Multiplexer Based Design for Ternary Logic Circuits // IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia). 19—21 December 2013. Visakhapatnam, India. P. 46—51. DOI: 10.1109/PrimeAsia.2013.6731176.

14. **Ahmad S., Alam M.** Balanced-Ternary Logic for Improved and Advanced Computing // International Journal of Computer Science and Information Technologies (IJCSIT). 2014. Vol. 5, Iss. 4. P. 5157—5160.

15. **Гиниятуллин В. М., Арсланов И. Г., Богданова П. Д., Габитов Р. Н., Салихова М. А.** Способы реализации функций троичной логики // Программные системы и вычислительные методы. 2014. № 2. С. 239—254. DOI: 10.7256/2305-6061.2014.2.11820.

16. **Nair R. S. P., Smith S. C., Di J.** Delay Insensitive Ternary CMOS Logic for Secure Hardware // Journal of Low Power Electronics and Applications. 2015. Iss. 5. P. 183—215. DOI: 10.3390/jlpea5030183.

17. **Uma Mahesh R. N., Sudeep J.** Design and Novel Approach for Ternary and Quaternary Logic Circuits // 2nd International Conference for Convergence in Technology (I2CT). 7—9 April 2017. Mumbai, India. P. 1224—1227. DOI: 10.1109/I2CT.2017.8226322.

18. **Kim S., Lim T., Kang S.** An Optimal Gate Design for the Synthesis of Ternary Logic Circuits // 23rd Asia and South Pacific Design Automation Conference (ASP-DAC). 22—25 January 2018. Jeju, South Korea. P. 476—481. DOI: 10.1109/ASP-DAC.2018.8297369.

19. **Vudadha C., Rajagopalan S., Dusi A., Phaneendra P. S., Srinivas M. B.** Encoder-Based Optimization of CNFET-Based Ternary Logic Circuits // IEEE Transactions on Nanotechnology. 2018. Vol. 17. Iss. 2. P. 299—310. DOI: 10.1109/TNANO.2018.2800015.

20. **Rahman Md. R., Rise J. E.** On Designing a Ternary Reversible Circuit for Online Testability // IEEE Pacific Rim Conference on Communications, Computers and Signal Processing. 2011. P. 1—7. DOI: 10.1109/PACRIM.2011.6032878.

21. **Nayeem N. M., Rice J. E.** Design of an Online Testable Ternary Circuit from the Truth Table // Lecture Notes in Computer Science book series (LNCS, volume 7581): Reversible Computation, 4th International Workshop on Reversible Computation (RC 2012). Copenhagen, Denmark, July 2—3. 2012. P. 152—159.

22. **Пархоменко П. П., Согомонян Е. С.** Основы технической диагностики (оптимизация алгоритмов диагностирования, аппаратурные средства). М.: Энергоатомиздат, 1981. 320 с.

23. **Göessel M., Ocheretny V., Sogomonyan E., Marienfeld D.** New Methods of Concurrent Checking: Edition 1. Dordrecht: Springer Science + Business Media B. V., 2008. 184 p.

24. **Kharchenko V., Kondratenko Yu., Kacprzyk J.** Green IT Engineering: Concepts, Models, Complex Systems Architectures // Springer Book series "Studies in Systems, Decision and Control". 2017. Vol. 74. 305 p.

25. **Согомонян Е. С., Слабаков Е. В.** Самопроверяемые устройства и отказоустойчивые системы. М.: Радио и связь, 1989. 208 с.

26. **Siewiorek D., Swarz R.** Reliable Computer Systems: Design and Evaluation. USA, Bedford: Digital Press, 1992. 908 p.

27. **Goessel M., Graf S.** Error Detection Circuits. London: McGraw-Hill, 1994. 261 p.

28. **Mitra S., McCluskey E. J.** Which Concurrent Error Detection Scheme to Choose? // Proceedings of International Test Conference, 2000, USA, Atlantic City, NJ, 03—05 October 2000. P. 985—994. DOI: 10.1109/TEST.2000.894311.

29. **Brouwer A. E., Hamalainen H. O., Ostergard P. R. J., Sloane N. J. A.** Bounds on Mixed Binary/Ternary Codes // IEEE Transactions on Information Theory. 1988. Vol. 44, Iss. 1. P. 140—161. DOI: 10.1109/18.651001.

30. **Gulliver T. A., Ostergard P. R. J.** Improved Bounds for Ternary Linear Codes of Dimension 7 // IEEE Transactions on Information Theory. 1997. Vol. 43, Iss. 4. P. 1377—1381. DOI: 10.1109/18.605613.

31. **Bitouze N., Graell i Amat A., Rosnes E.** Error Correcting Coding for a Nonsymmetric Ternary Channel // IEEE Transactions on Information Theory. 2010. Vol. 56, Iss. 11. P. 5715—5729. DOI: 10.1109/TIT.2010.2069211.

32. **Laaksonen A., Östergård P. R. J.** New Lower Bounds on Error-Correcting Ternary, Quaternary and Quinary Codes // Lecture Notes in Computer Science 10495, Springer: Coding Theory and Applications. 5th International Castle Meeting, ICMCTA 2017, Vihula, Estonia, August 28—31, 2017. P. 228—237.

33. **Svanström M.** A Lower Bound for Ternary Constant Weight Codes // IEEE Transactions on Information Theory. 1997. Vol. 43. P. 1630—1632.

34. **Svanström M., Östergård P. R. J., Bogdanova G. T.** Bounds and Constructions for Ternary Constant-Composition Codes // IEEE Transactions on Information Theory. 2002. Vol. 48. P. 101—111.

35. **Поспелов Д. А.** Логические методы анализа и синтеза схем. М.: Энергия, 1974. 368 с.

36. **Abramovici M., Breuer M. A., Friedman A. D.** Digital System Testing and Testable Design. Computer Science Press, 1998. 652 p.

37. **Ефанов Д. В., Сапожников В. В., Сапожников Вл. В.** Синтез самопроверяемых комбинационных устройств на основе выделения специальных групп выходов // Автоматика и телемеханика. 2018. № 9. С. 79—94.

38. **Сапожников В. В., Сапожников Вл. В., Ефанов Д. В.** Классификация ошибок в информационных векторах систематических кодов // Известия вузов. Приборостроение. 2015. Т. 58, № 5. С. 333—343. DOI: 10.17586/0021-3454-2015-58-5-333-343.

39. **Сагалович Ю. Л.** Введение в алгебраические коды. М.: Ин-т проблем передачи информации им. А. А. Харкевича РАН, 2010. 302 с.

40. **Mirzaee R. F., Daliri M. S., Navi K., Bagherzadeh N.** A Single Parity-Check Digit for One Trit Error Detection in Ternary Communication Systems: Gate-Level and Transistor-Level Designs // Journal of multiple-valued logic and soft computing. 2017. Vol. 29, N. 3—4. P. 303—326.

D. V. Efanov, D. Sc., Associate Professor,
Head of the Direction of monitoring and diagnosis systems department of "LocoTech-Signal" LLC,
Professor of Russian university of transport, Moscow, Russian Federation, e-mail: TrES-4b@yandex.ru

The Ternary Parity Code in the On-Line Testing of Automation Devices

The author of the article raises the question about the development of methods for the on-line testing of ternary automation devices. The article presents an analogy between the approaches to the organization of systems for devices on-line testing using binary and ternary logic. The ternary parity code and some of its properties are described, the accounting of which is expedient when organizing diagnostic systems. It is proved that all parity code data vectors are distributed evenly among all its check vectors. A formula for counting the number of undetectable errors in the ternary code data vectors with a uniform distribution of all data vectors between all check vectors is given. A coding theorem is proved with the least total number of undetectable errors in the ternary code data vectors for fixed values of the lengths of data and check vectors. The rules for constructing modified ternary parity codes are given.

Keyword: *automation and computer engineering; ternary logic; on-line testing; the calculated result; indirectly controlled; errors at the outputs; ternary parity code*

DOI: 10.17587/it.25.426-434

References

1. **Aspray W.** John von Neumann and the Origins of Modern Computing (History of Computing), Boston, Cambridge, MIT, 1990, 396 p.

2. **Karpenko A. S.** Razvitie mnogoznachnoj logiki (The development of multi-valued logic), Moscow, Pub. House LKI, 2010, 448 p. (in Russian).

3. **Brusencov N. P.** *Istoriko-matematicheskie issledovaniya; Pod redakciej S. S. Demidova. Vtoraya seriya*, iss. 11 (46), Moscow, Yanus-K, 2006, pp. 228—234 (in Russian).

4. **Brusencov N. P., Maslov S. P., Rozin V. P., Tishulina A. M.** Small digital computing machine Setun, Moscow, Pub. House of MGU, 1962, 140 p. (in Russian).

5. **Connely J.** Ternary Computing Testbed 3-Trit Computer Architecture, California Polytechnic State University of San Luis Obispo, August 29th, 2008, 184 p.

6. **Roy D., Merril Jr.** *Proceedings of the SHARE design automation project (DAC '65),* ACM New York, NY, USA, pp. 6.1—6.17, DOI: 10.1145/800266.810759.

7. **Hu M., Smith K. C.** *Canadian Electrical Engineering Journal*, 1984, vol. 9, iss. 3, pp. 100—104, DOI: 10.1109/CEEJ.1984.6593793.

8. **Wu J.** *Proceedings of 19th International Symposium on Multiple-Valued Logic*, 29—31 May 1989, Guangzhou, China, pp. 94—99, DOI: 10.1109/ISMVL.1989.37766.

9. **Lanyon B. P., Barbieri M., Almeida M. P., Jennewein T., Ralph T. C., Resch K. J., Pryde G. J., O'Brien J. L., Gilchrist A., White A. G.** *Nature Physics*, 2009, vol. 5, iss. 2, pp. 134—140, DOI: 10.1038/nphys1150.

10. **Cambou B., Flikkema P. G., Palmer J., Telesca D., Philabaum C.** Cryptography, 2018, vol. 2, iss. 1 (March 2018), pp. 1—16, DOI: 10.3390/cryptography2010006.

11. **Petrov A.** *Populyarnaya mekhanika*, 2011, iss. 9, pp. 72—76 (in Russian).

12. **Hahanov V.** Cyber Physical Computing for IoT-driven Services, New York, Springer International Publishing AG, 2018, 279 p.

13. **Vudadha C., Katragadda S., Phaneendra P. S.** *IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, 19—21 December 2013, Visakhapatnam, India, pp. 46—51, DOI: 10.1109/PrimeAsia.2013.6731176.

14. **Ahmad S., Alam M.** *International Journal of Computer Science and Information Technologies (IJCSIT),* 2014, vol. 5, iss. 4, pp. 5157—5160.

15. **Giniyatullin V. M., Arslanov I. G., Bogdanova P. D., Gabitov R. N., Salihova M. A.** *Programmnye Sistemy i Vychislitel'nye Metody*, 2014, iss. 2, pp. 239—254, DOI: 10.7256/2305-6061.2014.2.11820 (in Russian).

16. **Nair R. S.P., Smith S. C., Di J.** *Journal of Low Power Electronics and Applications,* 2015, iss. 5, pp. 183—215, DOI: 10.3390/jlpea5030183.

17. **Uma Mahesh R. N., Sudeep J.** *2nd International Conference for Convergence in Technology (I2CT),* 7—9 April 2017, Mumbai, India, pp. 1224—1227, DOI: 10.1109/I2CT.2017.8226322.

18. **Kim S., Lim T., Kang S.** *23rd Asia and South Pacific Design Automation Conference (ASP-DAC),* 22—25 January 2018, Jeju, South Korea, pp. 476—481, DOI: 10.1109/ASPDAC.2018.8297369.

19. **Vudadha C., Rajagopalan S., Dusi A., Phaneendra P. S., Srinivas M. B.** *IEEE Transactions on Nanotechnology*, 2018, vol. 17, iss. 2, pp. 299—310, DOI: 10.1109/TNANO.2018.2800015.

20. **Rahman Md.R., Rise J. E.** *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2011, pp. 1—7, DOI: 10.1109/PACRIM.2011.6032878.

21. **Nayeem N. M., Rice J. E.** *Lecture Notes in Computer Science book series (LNCS, volume 7581): Reversible Computation, 4th International Workshop on Reversible Computation (RC 2012),* Copenhagen, Denmark, July 2—3, 2012, pp. 152—159.

22. **Parkhomenko P. P., Sogomonyan E. S.** Technical Diagnosis Fundamentals (Diagnostic Algorithm Optimization, Apparatus Means), Moscow, Energoatomizdat, 1981, 320 p. (in Russian).

23. **Göessel M., Ocheretny V., Sogomonyan E., Marienfeld D.** New Methods of Concurrent Checking: Edition 1, Dordrecht: Springer Science + Business Media B. V., 2008, 184 p.

24. **Kharchenko V., Kondratenko Yu., Kacprzyk J.** Green IT Engineering: Concepts, Models, Complex Systems Architectures, *Springer Book series "Studies in Systems, Decision and Control"*, vol. 74, 2017, 305 p.

25. **Sogomonyan E. S., Slabakov E. V.** Self-checking devices and fault-tolerant systems, Radio & Svjaz`, Moscow, 208 p. (in Russian).

26. **Siewiorek D., Swarz R.** Reliable Computer Systems: Design and Evaluation, USA, Bedford, Digital Press, 1992, 908 p.

27. **Goessel M., Graf S.** Error Detection Circuits, London, McGraw-Hill, 1994, 261 p.

28. **Mitra S., McCluskey E. J.** *Proceedings of International Test Conference*, 2000, USA, Atlantic City, NJ, 03—05 October 2000, pp. 985—994, DOI: 10.1109/TEST.2000.894311.

29. **Brouwer A. E., Hamalainen H. O., Ostergard P. R. J., Sloane N. J. A.** Bounds on Mixed Binary/Ternary Codes // IEEE Transactions on Information Theory, 1988, vol. 44, iss. 1, pp. 140—161, DOI: 10.1109/18.651001.

30. **Gulliver T. A., Ostergard P. R.J.** *IEEE Transactions on Information Theory*, 1997, vol. 43, iss. 4, pp. 1377—1381, DOI: 10.1109/18.605613.

31. **Bitouze N., Graell i Amat A., Rosnes E.** *IEEE Transactions on Information Theory*, 2010, vol. 56, iss. 11, pp. 5715—5729, DOI: 10.1109/TIT.2010.2069211.

32. **Laaksonen A., Östergård P. R.J.** *Lecture Notes in Computer Science 10495, Springer: Coding Theory and Applications. 5th International Castle Meeting, ICMCTA 2017*, Vihula, Estonia, August 28—31, 2017, pp. 228—237.

33. **Svanström M.** A *IEEE Transactions on Information Theory*, 1997, vol. 43, pp. 1630—1632.

34. **Svanström M., Östergård P. R. J., Bogdanova G. T.** *IEEE Transactions on Information Theory*, 2002, vol. 48, pp. 101—111.

35. **Pospelov D. A.** Logical methods of analysis and synthesis of circuits, Moscow, Energija, 1974, 368 p. (in Russian).

36. **Abramovici M., Breuer M. A., Friedman A. D.** Digital System Testing and Testable Design, Computer Science Press, 1998, 652 p.

37. **Efanov D. V., Sapozhnikov V. V., Sapozhnikov Vl. V.** *Avtomatika i telemekhanika*, 2018, iss. 9, pp. 79—94. (in Russian).

38. **Sapozhnikov V. V., Sapozhnikov Vl. V., Efanov D. V.** *Izvestiya Vysshikh Uchebnykh Zavedeniy. Priborostroenie*, 2015, vol. 58, iss. 5, pp. 333—343 (in Russian).

39. **Sagalovich Yu. L.** Introduction to Algebraic Codes, Moscow, Institut problem peredachi informacii im. A. A. Harkevicha RAN, 2010, 302 p. (in Russian).

40. **Mirzaee R. F., Daliri M. S., Navi K., Bagherzadeh N.** *Journal of multiple-valued logic and soft computing*, 2017, 29 (3—4), pp. 303—326.