ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2019 Управление, вычислительная техника и информатика № 48

ПРОЕКТИРОВАНИЕ И ДИАГНОСТИКА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

УДК 004.312
DOI: 10.17223/19988605/48/11

Н.Б. Буторина, Е.Г. Пахомова

О ДЛИНАХ ПУТЕЙ САМОТЕСТИРУЕМЫХ ДЕТЕКТОРОВ, ПОСТРОЕННЫХ В БАЗЕ ПЛБ

Исследование выполнено при поддержке гранта Российского научного фонда, проект № 14-19-00218.

При проектировании самопроверяемых схем используются самотестируемые детекторы кодов, в частности детекторы равновесных кодов ((*m*, *n*)-кодов). Для представления всевозможных кодовых слов (*m*, *n*)-кодов используется специальная формула разложения. В работе сравниваются два самотестируемых детектора (*m*, *n*)-кодов, построенные на многократном использовании этой формулы разложения и реализации полученной в результате формулы программируемыми логическими блоками (ПЛБ). Конкретный вид формулы определяется множеством представляемых ею кодовых слов (параметрами *m*, *n*), числом *k* входов в ПЛБ и способом разбиения переменных на подмножества. В работе анализируется влияние способа разбиения переменных на подмножества на максимальную длину пути в схеме детектора и на количество ПЛБ в детекторе.
Ключевае слова: самотестируемость; детектор; программируемые логические блоки; равновесный код.

В самопроверяемых схемах используются детекторы кодов. Как правило, детектор строится на той же элементной базе, что и схема. Метод построения детектора обычно состоит в подсчете веса входного кодового слова. Для этой цели используются схемы, основанные либо на пороговых элементах, либо на параллельных счетчиках [1–7]. В данной работе предлагается синтез детектора на ПЛБ. В [8] был предложен метод проектирования самопроверяемого конечного автомата. В случае исправного функционирования схемы на выходе комбинационной составляющей реализуются кодовые слова некоторого неупорядоченного кода, например равновесного. Там же предлагается метод синтеза самотестируемого детектора равновесного кода (детектора (*m*, *n*)-кодов), основанный на многократном использовании специальной формулы разложения множества кодовых слов равновесного кода и реализации полученной в результате формулы программируемыми логическими блоками в предположении, что среди программируемых блоков обязательно присутствуют блоки с двумя выходами. К реализации формулы предъявляется специальное требование: соответствующая реализации схема детектора (*m*, *n*)-кодов должна быть самотестируемой для заданного множества *V* неисправностей. Предлагаемый для сравнения подход [9] основан на многократном использовании той же формулы разложения, применяемой к иному, по сравнению с работой [8], разбиению подмножества переменных кодовых слов. Это разбиение ориентировано на сокращение максимальной длины путей в схеме, являющейся реализацией полученной в результате формулы, и минимизацию разброса длин путей. В предложенном в работе [9] методе синтеза самотестируемых детекторов наличие двух выходных ПЛБ не обязательно. Это значит, что данный метод позволяет строить самотестируемые детекторы, используя любые существующие на текущий момент программируемые логические матрицы (Field Programmable Gate Arrays (FPGA), производители Xilinx, Altera, Achronix, Actel, Atmel, Lattice semiconductor и др.), в то время как метод, предложенный в [8], ориентирован только на FPGA фирмы Xilinx.

Рассматривается множество V неисправностей детектора, которое включает в себя все кратные константные неисправности на входах и выходах ПЛБ. При этом в схеме детектора неисправным может быть только один ПЛБ. Предполагается, что в системе, состоящей из самопроверяемой схемы и детектора (m, n)-кодов, неисправными могут быть либо схема, либо детектор, но не оба вместе.

К самотестируемому детектору предъявляются следующие требования:

1) при появлении на выходе схемы (или на входе детектора) в некоторый момент времени t некодового слова детектор должен выдать соответствующий сигнал;

2) в самом детекторе может произойти неисправность из рассматриваемого множества неисправностей V, которая должна быть обнаружима в рабочей области функционирования детектора, т.е. на множестве всех его кодовых слов. Это означает, что должен существовать кодовый набор (кодовое слово (m, n)-кода), на котором эта неисправность проявляется на выходах детектора.

Самотестируемый детектор имеет два выхода, причем комбинации значений сигналов имеют следующие интерпретации:

а) (01) или (10) означает, что входной набор является кодовым словом ((m, n)-кода) и детектор исправен;

б) (00) или (11) означает, что либо входной набор не является кодовым словом, либо детектор неисправен.

Детектор строится по формуле, представляющей множество всех кодовых слов (m, n)-кода, являясь реализацией этой формулы.

1. Метод разложения множества кодовых слов (m, n)-кода

Заметим, что число всевозможных кодовых слов (m, n)-кода равно C_n^m, т.е. числу сочетаний из n по m. Кодовые слова могут быть представлены дизъюнкцией элементарных конъюнкций (ДНФ) ранга n. Обозначим эту дизъюнкцию $D_n^m(X)$, где $X = \{x_1, \ldots, x_n\}$ – множество переменных. Уже при $n = 10$, $m = 5$ ДНФ $D_{10}^5(X)$ состоит из $C_{10}^5 = 252$ конъюнкций ранга 10 и содержит 2 520 букв. Поскольку любые две конъюнкции из $D_n^m(X)$ ортогональны по крайней мере по двум переменным, то ДНФ $D_n^m(X)$ является совершенной и сокращенной одновременно и, следовательно, не может быть сокращена в результате минимизации ДНФ.

Для представления всевозможных (m, n)-кодов в [8] предложена специальная формула, включающая скобки, символы $\wedge$, $\vee$ и ДНФ $D_p^q(X^r)$. Выражение $D_p^q(X^r)$ – это дизъюнкция конъюнкций, соответствующих всем (q, p)-кодовым словам, $p \le n$, $q \le p$, $X^r \subset X$, $X = \{x_1, \ldots, x_n\}$. Построим формулу следующим образом.

Разделим множество X на два подмножества X^1, X^2, где $X^1 = \{x_1, \ldots, x_k\}$, $X^2 = \{x_{k+1}, \ldots, x_n\}$.

Все множество кодовых слов равновесного кода может быть представлено формулой

$$D_n^m(X) = \sum_{i=0}^{m} D_k^i(X^1) D_{n-k}^{m-i}(X^2), \qquad (1)$$

где символ $\wedge$ между $D_k^i(X^1)$ и $D_{n-k}^{m-i}(X^2)$ опущен.

Назовем k основой разложения (k – число входов в ПЛБ), D_k^i, D_{n-k}^{m-i} – функциями разложения. Если $n - k > k$, то формула (1) снова используется для соответствующей функции разложения D_{n-k}^{m-i}, $i = \overline{0, m}$, и т.д.

В результате многократного применения формулы (1) получаем представление для всех кодовых слов (m, n)-кода в виде формулы, содержащей скобки, операции $\vee$ и $\wedge$ и $D_p^q(X^r)$. В этой формуле для любой ДНФ $D_p^q(X^r)$ выполняется условие $p \le k$. Представления такого вида в дальнейшем будем называть *формулами А*.

Например, получим формулу A для D_8^4, k = 2 при помощи формулы (1).

В соответствии с приведенным алгоритмом на первом шаге: $X^1 = \{x_1, x_2\}$, $X^2 = \{x_3, x_4, x_5, x_6, x_7, x_8\}$.

$$D_8^4(X) = D_2^0(X^1)D_6^4(X^2) \vee D_2^1(X^1)D_6^3(X^2) \vee D_2^2(X^1)D_6^2(X^2).$$

Далее, $X^{21} = \{x_3, x_4\}$, $X^{22} = \{x_5, x_6, x_7, x_8\}$. Выполнив следующий шаг разложения по формуле (1) для $D_6^2(X^2)$, $D_6^3(X^2)$, $D_6^4(X^2)$ получим

$$D_6^2(X^2) = D_2^0(X^{21})D_4^2(X^{22}) \vee D_2^1(X^{21})D_4^1(X^{22}) \vee D_2^2(X^{21})D_4^0(X^{22}),$$
$$D_6^3(X^2) = D_2^0(X^{21})D_4^3(X^{22}) \vee D_2^1(X^{21})D_4^2(X^{22}) \vee D_2^2(X^{21})D_4^1(X^{22}),$$
$$D_6^4(X^2) = D_2^0(X^{21})D_4^4(X^{22}) \vee D_2^1(X^{21})D_4^3(X^{22}) \vee D_2^2(X^{21})D_4^2(X^{22}).$$

Далее, $X^{221} = \{x_5, x_6\}$, $X^{222} = \{x_7, x_8\}$. Выполнив следующий шаг разложения по формуле (1) для $D_4^0(X^{22})$, $D_4^1(X^{22})$, $D_4^2(X^{22})$, $D_4^3(X^{22})$, $D_4^4(X^{22})$ получим

$$D_4^0(X^{22}) = D_2^0(X^{221})D_2^0(X^{222}),$$
$$D_4^1(X^{22}) = D_2^0(X^{221})D_2^1(X^{222}) \vee D_2^1(X^{221})D_2^0(X^{222}),$$
$$D_4^2(X^{22}) = D_2^0(X^{221})D_2^2(X^{222}) \vee D_2^1(X^{221})D_2^1(X^{222}) \vee D_2^2(X^{221})D_2^0(X^{222}),$$
$$D_4^3(X^{22}) = D_2^1(X^{221})D_2^2(X^{222}) \vee D_2^2(X^{221})D_2^1(X^{222}),$$
$$D_4^4(X^{22}) = D_2^2(X^{221})D_2^2(X^{222}).$$

Подставляя полученные выражения в формулу (1) получаем формулу A. Структура этой формулы может быть представлена деревом (рис. 1).

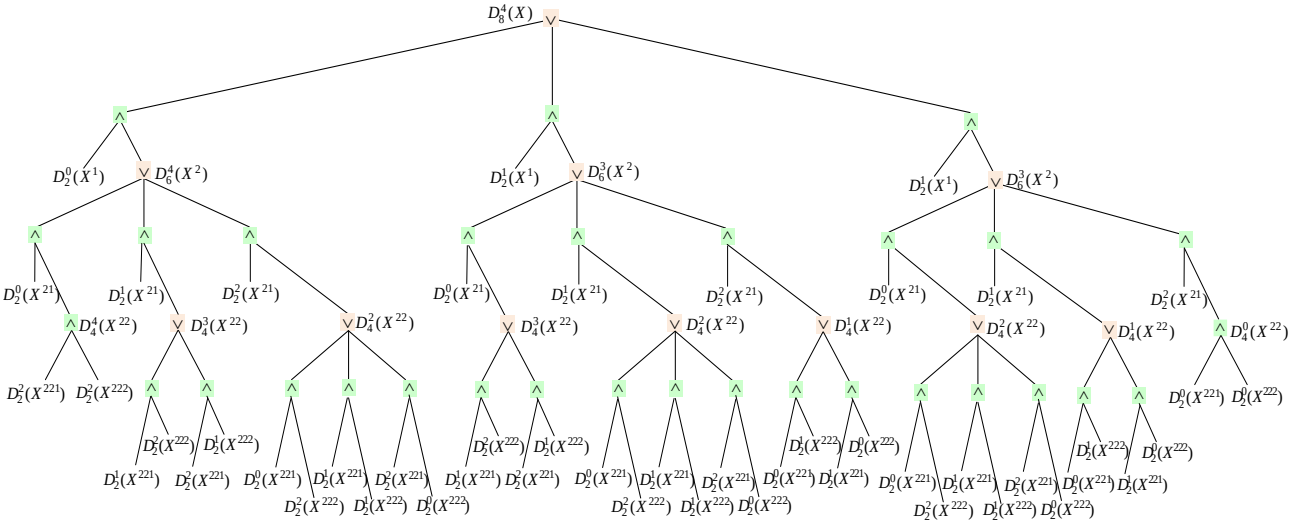

Рис. 1. Дерево разложения для D_8^4, $k = 2$ (способ 1)

Fig. 1. Tree decomposition for D_8^4, $k = 2$ (technique 1)

В [9] был предложен другой способ разбиения множества переменных на подмножества.

Разделим множество X на два подмножества X^1, X^2, где $X^1 = \{x_1, \ldots, x_g\}$, $X^2 = \{x_{g+1}, \ldots, x_n\}$. Все множество кодовых слов равновесного кода может быть представлено формулой

$$D_n^m(X) = \sum_{i=0}^{m} D_g^i(X^1)D_{n-g}^{m-i}(X^2), \qquad (2)$$

где символ $\wedge$ между $D_k^i(X^1)$ и $D_{n-k}^{m-i}(X^2)$ опущен.

Здесь $D_g^i(X^1)$, $D_{n-g}^{m-i}(X^2)$ – функции разложения, основа разложения k – максимальная мощность подмножества переменных при последнем применении формулы (2).

В статье [9] предлагается следующий способ выбора g: g – наименьшее целое, большее или равное числа $n/2$ или, в другой записи, $g = \lceil n/2 \rceil$. Если $g > k$ и / или $n - g > k$, то формула (2) снова используется для каждой функции разложения $D_g^i(X^1)$, $D_{n-g}^{m-i}(X^2)$, $i = \overline{0,m}$, и т.д. В результате мы имеем формулу для всех кодовых слов (m, n)-кода, в которой для любой $D_p^q(X^r)$ выполняется условие $p \le k$. Это значит, что во втором способе многократное разложение применяется как к первому, так и ко второму множителю формулы (2).

Например, получим формулу A для D_8^4, $k = 2$ вторым способом.

На первом шаге $X^1 = \{x_1, x_2, x_3, x_4\}$, $X^2 = \{x_5, x_6, x_7, x_8\}$. Результат разложения представляется в виде:

$$D_8^4(X) = D_4^0(X^1)D_4^4(X^2) \vee D_4^1(X^1)D_4^3(X^2) \vee D_4^2(X^1)D_4^2(X^2) \vee D_4^3(X^1)D_4^1(X^2) \vee D_4^4(X^1)D_4^0(X^2).$$

На втором шаге имеем подмножества $X^{11} = \{x_1, x_2\}$, $X^{12} = \{x_3, x_4\}$, $X^{21} = \{x_5, x_6\}$, $X^{22} = \{x_7, x_8\}$. Применим формулу (2) еще раз ко всем функция разложения:

$$D_4^0(X^1) = D_2^0(X^{11})D_2^0(X^{12}),$$
$$D_4^1(X^1) = D_2^0(X^{11})D_2^1(X^{12}) \vee D_2^1(X^{11})D_2^0(X^{12}),$$
$$D_4^2(X^1) = D_2^0(X^{11})D_2^2(X^{12}) \vee D_2^1(X^{11})D_2^1(X^{12}) \vee D_2^2(X^{11})D_2^0(X^{12}),$$
$$D_4^3(X^1) = D_2^1(X^{11})D_2^2(X^{12}) \vee D_2^2(X^{11})D_2^1(X^{12}),$$
$$D_4^4(X^1) = D_2^2(X^{11})D_2^2(X^{12}),$$
$$D_4^0(X^2) = D_2^0(X^{21})D_2^0(X^{22}),$$
$$D_4^1(X^2) = D_2^0(X^{21})D_2^1(X^{22}) \vee D_2^1(X^{21})D_2^0(X^{22}),$$
$$D_4^2(X^2) = D_2^0(X^{21})D_2^2(X^{22}) \vee D_2^1(X^{21})D_2^1(X^{22}) \vee D_2^2(X^{21})D_2^0(X^{22}),$$
$$D_4^3(X^2) = D_2^1(X^{21})D_2^2(X^{22}) \vee D_2^2(X^{21})D_2^1(X^{22}),$$
$$D_4^4(X^2) = D_2^2(X^{21})D_2^2(X^{22}).$$

Подставляя полученные выражения в формулу (2) получаем формулу A.

Можно сказать, что формула (1) есть частный случай формулы (2) при $g = k$.

Структура формулы (2) может быть представлена деревом (рис. 2).

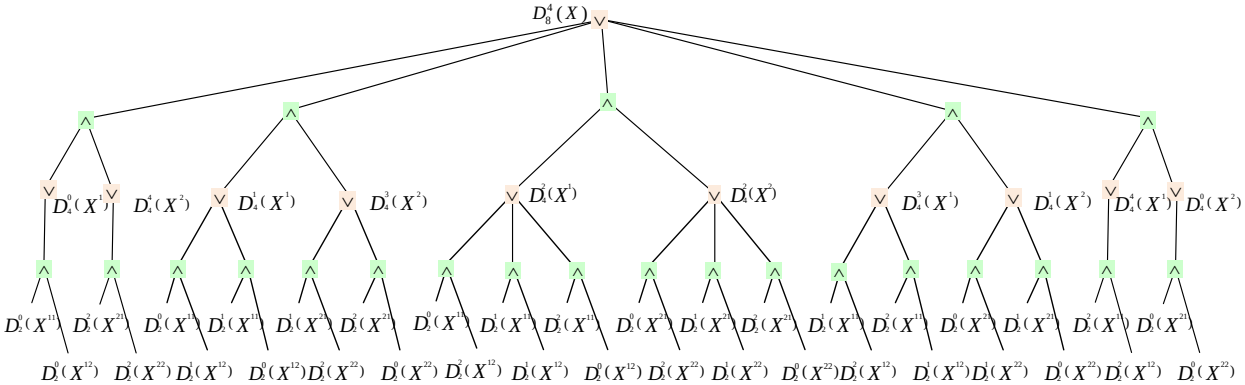

Рис. 2. Дерево разложения для D_8^4, $k = 2$ (способ 2)

Fig. 2. Tree decomposition for D_8^4, $k = 2$ (technique 2)

Дадим определение шага разложения.

1. При первом применении формулы (2) множество переменных X разбивается на два подмножества: X^1 и X^2. При этом мощность одного из этих подмножеств или обоих подмножеств может оказаться больше k. Это первый шаг разложения.

2. На втором шаге разложения формула (2) применяется ко всем функциям разложения предыдущего шага, определенным на подмножестве переменных, мощность которого больше k. В результате применения формулы (2) подмножество X^1 и / или X^2 разбивается на два подмножества, мощность каждого из которых меньше мощности X^1 или X^2 соответственно, но при этом снова может быть больше k.

3. Таким образом, i-й шаг разложения – это применение формулы (2) ко всем функциям разложения предыдущего $(i-1)$-го шага, определенным на подмножестве переменных, мощность которого больше k.

Итак, конкретный вид формулы определяется множеством представляемых ею кодовых слов (параметрами m, n), значением k и способом разбиения переменных на подмножества.

В дальнейшем будем сравнивать рассмотренные выше способы разбиения множеств переменных на два подмножества. По сути, формула (1) есть частный случай формулы (2). Действительно, в первом способе разложения $g = k$ на каждом шаге разложения, и формула (1) используется только для функции разложения D_{n-k}^{m-i}, $i = \overline{0,m}$, и т.д.

Во втором способе выбирается g, кратное k. При этом возможны следующие две ситуации.

1. Пусть n кратно k, т.е. $n = kq = k(q^1 + q^2)$, где kq^1 – ближайшее к $\lceil n/2 \rceil$ число, кратное k, $q^2 = q - q^1$. Итак, на первом шаге множество X, $|X| = n$, разбивается на два подмножества X^1 и X^2, где $|X^1| = kq^1 = g$, $|X^2| = kq^2 = n - g$.

Если $|X^1| > k$ и (или) $|X^2| > k$, то применяем формулу (2) к соответствующему сомножителю $D_g^i(X^1)$, $D_{n-g}^{m-i}(X^2)$, выполняя разбиение множеств X^1, X^2 в соответствии с правилами пункта 1, и т.д.

2. Пусть n не кратно k. Тогда при использовании формулы (2) на первом шаге разложения предлагается выбирать g – ближайшее к $\lceil n/2 \rceil$ число, кратное k. В этом случае $n - g$ не кратно k. Итак, на первом шаге множество X, $|X| = n$, разбивается на два подмножества X^1 и X^2, причем $|X^1| = kq^1 = g$, $|X^2| = kq^2 + r = n - g$, где $r < k$.

Если $|X^1| > k$ и (или) $|X^2| > k$, то применяем формулу (2) к соответствующему сомножителю $D_g^i(X^1)$, $D_{n-g}^{m-i}(X^2)$, выполняя разбиение множеств X^1, X^2 в соответствии с правилами пункта 2, и т.д.

Приведем пример нахождения конкретных значений g и $n - g$ при $k = 6$ для $n = 18, 21, 30$ двумя способами (рис. 3, 4).

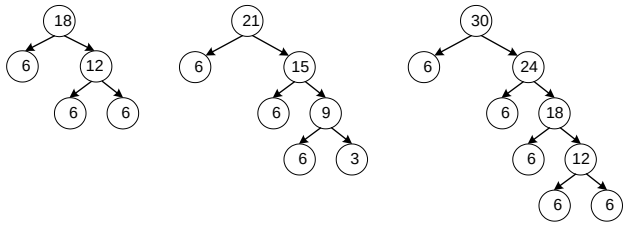

Рис.3. Деревья выбора при $k = 6$ для $n = 18, 21, 30$ (способ 1)
Fig. 3. Trees of choice with $k = 6$ for $n = 18, 21, 30$ (technique 1)

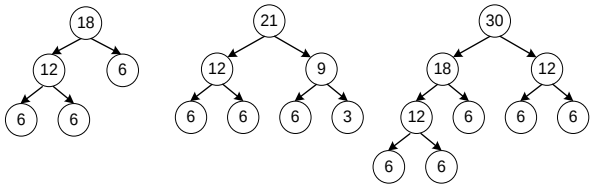

Рис.4. Деревья выбора при $k = 6$ для $n = 18, 21, 30$ (способ 2)
Fig. 4. Trees of choice with $k = 6$ for $n = 18, 21, 30$ (technique 2)

В обоих способах концевым вершинам дерева разложения сопоставляются ДНФ вида D_k^i, где $0 \le i \le k$.

Под синтезом детектора будем понимать реализацию формулы A композицией программируемых логических блоков, основанную на анализе дерева разложения формулы A. Подход к синтезу самотестируемого детектора для обоих способов остается одинаковым: вершины дерева разложения формулы A покрываются программируемыми логическими блоками снизу вверх [8].

Заметим, что кодовые слова, представляемые формулой A, при синтезе детектора необходимо разбивать на два подмножества: каждое из подмножеств сопоставляется одному из выходов детектора.

2. Сравнение двух детекторов

Проведем оценку детекторов по двум параметрам: количеству ПЛБ и длине пути.

Последовательность ПЛБ, такую что выход предыдущего блока является входом в последующий блок, назовем *путем* в детекторе. Входы первого блока являются входами детектора, а выход последнего блока – один из выходов детектора. *Длиной пути* назовем число ПЛБ, образующих путь. Нас будут интересовать пути максимальной длины, поскольку они влияют на быстродействие самопроверяемой системы, состоящей из самопроверяемой схемы и контролирующего ее самотестируемого детектора.

Договоримся детектор, построенный по формуле (1) называть *детектор* 1, а детектор, построенный по формуле (2) – *детектор* 2.

При $n \le 3k$ значения g и $n - g$ для обоих способов разложения одинаковы, следовательно, одинаковы формулы A и представляющие их деревья разложения. Различия появляются при $n > 3k$.

Проведем оценку детекторов по максимальной длине пути $L_{\max}$. Она зависит от числа μ шагов разложения и максимальной длины пути λ в подсхемах, реализующих формулу (1) или (2) на очередном шаге разложения.

Число μ шагов разложения зависит от выбранного способа разбиения переменных на подмножества.

Максимальная длина пути λ в подсхеме, реализующей формулу на очередном шаге разложения, зависит от количества слагаемых в этой формуле. Покажем это на примере.

Применим формулу (2) к D_{14}^7 при $k = 7$ с целью получения реализующей D_{14}^7 подсхемы:

$$D_{14}^7 = D_7^0(X^1)D_7^7(X^2) \vee D_7^1(X^1)D_7^6(X^2) \vee D_7^2(X^1)D_7^5(X^2) \vee D_7^3(X^1)D_7^4(X^2) \vee \\ \vee D_7^4(X^1)D_7^3(X^2) \vee D_7^5(X^1)D_7^2(X^2) \vee D_7^6(X^1)D_7^1(X^2) \vee D_7^7(X^1)D_7^0(X^2). \quad (3)$$

Обозначения выходов ПЛБ приведены в табл. 1.

Таблица 1

Обозначения выходов ПЛБ, реализующих множители формулы (3)

Реализуемая функция	Обозначение выхода	Реализуемая функция	Обозначение выхода
$D_7^0(X^1)$	y_1	$D_7^0(X^2)$	y_{16}
$D_7^1(X^1)$	y_3	$D_7^1(X^2)$	y_{14}
$D_7^2(X^1)$	y_5	$D_7^2(X^2)$	y_{12}
$D_7^3(X^1)$	y_7	$D_7^3(X^2)$	y_{10}
$D_7^4(X^1)$	y_9	$D_7^4(X^2)$	y_8
$D_7^5(X^1)$	y_{11}	$D_7^5(X^2)$	y_6
$D_7^6(X^1)$	y_{13}	$D_7^6(X^2)$	y_4
$D_7^7(X^1)$	y_{15}	$D_7^7(X^2)$	y_2

Имеем 8 слагаемых, состоящих из двух множителей. Каждый множитель сопоставляется входу ПЛБ, который реализует дизъюнкцию нескольких слагаемых. Так как в данном примере количество входов ПЛБ равно 7, то для реализации всех слагаемых потребуется $\lceil 2 \cdot 8/7 \rceil = 3$ ПЛБ. Обозначим выходы этих ПЛБ через z_1, z_2, z_3.

Таким образом, ПЛБ1 реализует функцию $y_1y_2 \vee y_3y_4 \vee y_6y_7$, выход этого ПЛБ обозначен через z_1, ПЛБ2 реализует функцию $y_7y_8 \vee y_9y_{10} \vee y_{11}y_{12}$, выход этого ПЛБ обозначен через z_2, ПЛБ3 реализует функцию $y_{13}y_{14} \vee y_{15}y_{16}$, выход этого ПЛБ обозначен через z_3.

Пусть выходы подсхемы, реализующей формулу (3), являются выходами детектора (их два). Выходы z_1 и z_2 делаем входами ПЛБ, реализующий функцию D_2^1. Выход этого ПЛБ является одним из выходов детектора, второй выход – z_3. Максимальная длина пути $\lambda_{\max}$ в подсхеме, реализующей формулу (3), равна 2. Эта подсхема приведена на рис. 5.

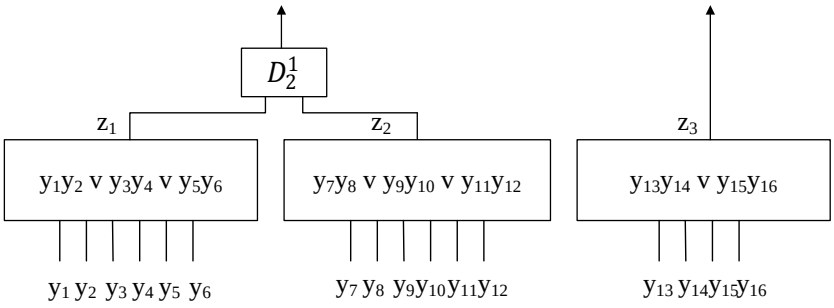

Рис. 5. Подсхема, реализующая формулу (3)
Fig. 5. A subcircuit realizing formula (3)

В общем случае, если на реализацию формулы (1) или (2) требуется два и более ПЛБ и выходы этих ПЛБ не являются выходами детектора, то максимальная длина пути $\lambda_{\max}$ в подсхеме, реализующей формулу, больше 1.

Пусть $L_{\max}$ – максимальная длина пути в схеме, реализующей детектор, $\lambda^i_{\max}$ – максимальная длина пути среди всех подсхем, реализующих i-й шаг разложения с использованием формулы (1) или (2). Тогда по построению $L_{\max} = \sum_{i=1}^{\mu} \lambda^i_{\max} + 1$, т.е. $L_{\max}$ – сумма всех максимальных длин путей на каждом шаге разложения + 1 (для реализации листьев дерева разложения).

Чтобы найти длину пути λ в подсхеме, реализующей формулу (2) или (1) на очередном шаге разложения, необходимо знать количество слагаемых *sum* в этой формуле.

Утверждение 1. Количество слагаемых *sum* в формуле (2) зависит от m, $n-m$, g и $n-g$, а именно $sum = \min\{m + 1, g + 1, n - g + 1, n - m + 1\}$.

Доказательство. Формула (2) имеет вид $D_n^m(X) = \sum_{i=0}^{m} D_g^i(X^1) D_{n-g}^{m-i}(X^2)$. Разберем всевозможные случаи и приведем примеры.

1. Пусть $m > g$, $m < n - g$. Так как в разложении D_g^i индекс i начинается с 0 и не может быть больше g, то получаем в формуле (2) $(g + 1)$-е слагаемое. Рассматривая связь чисел m, g и $n - g$, получаем, что в этом случае

$$g + 1 = \min\{g + 1, m + 1, n - g + 1, n - m + 1\}.$$

Действительно, условия $m > g$ и $m < n - g$ можно переписать в виде двойного неравенства $g < m < n - g$. Откуда находим

$$g + 1 < m + 1 < n - g + 1.$$

Иначе говоря,

$$g + 1 = \min\{g + 1, m + 1, n - g + 1\}.$$

Кроме того, из $m < n - g$ следует, что $n - m > g$, следовательно, $n - m + 1 > g + 1$. Таким образом,

$$g + 1 = \min\{g + 1, m + 1, n - g + 1, n - m + 1\}.$$

Например, пусть $g = 4$, $m = 5$ и $n = 10$. Тогда формула (2) имеет вид:

$$D_{10}^5 = D_4^0(X^1) D_6^5(X^2) \vee D_4^1(X^1) D_6^4(X^2) \vee D_4^2(X^1) D_6^3(X^2) \vee D_4^3(X^1) D_6^2(X^2) \vee D_4^4(X^1) D_6^1(X^2)$$

То есть количество слагаемых $sum = 5 = g + 1$.

2. Пусть $m < g$, $m > n - g$. В этом случае слагаемых $n - g + 1$.

Поскольку из неравенств $m < g$, $m > n - g$ следует, что $n - g + 1 < m + 1 < g + 1$, то $n - g + 1 = \min\{n - g + 1, m + 1, g + 1\}$. Кроме того, из $m < g$ находим: $-m > -g$, следовательно

$$n - m + 1 > n - g + 1.$$

Таким образом, $n - g + 1 = \min\{g + 1, m + 1, n - g + 1, n - m + 1\}$.

Например, пусть $g = 6$, $m = 4$ и $n = 9$. Тогда формула (2) имеет вид:

$$D_9^4 = D_6^1(X^1) D_3^3(X^2) \vee D_6^2(X^1) D_3^2(X^2) \vee D_6^3(X^1) D_3^1(X^2) \vee D_6^4(X^1) D_3^0(X^2).$$

То есть количество слагаемых $sum = 4 = n - g + 1$.

3. Пусть $m < g$, $m < n - g$. В этом случае слагаемых $m + 1 = \min\{m + 1, g + 1, n - g + 1\}$. Кроме того, из $m < g$, $m < n - g$, получаем: $m < g < n - m \Rightarrow m < n - m$, а значит $m + 1 < n - m + 1$.

Следовательно, $m + 1 = \min\{m + 1, g + 1, n - g + 1, n - m + 1\}$.

Например, пусть $g = 4$, $m = 2$ и $n = 8$. Тогда формула (2) имеет вид:

$$D_8^2 = D_4^0(X^1)D_4^2(X^2) \vee D_4^1(X^1)D_4^1(X^2) \vee D_4^2(X^1)D_4^0(X^2).$$

То есть количество слагаемых $sum = 3 = m + 1$.

4. Пусть $m > g$, $m > n - g$. В этом случае i в формуле (2) должно удовлетворять условиям $i \le g$ и $m - i \le n - g$. Откуда находим, что $m - n + g \le i \le g$. Это неравенство имеет $g - (m - n + g) + 1 = n - m + 1$ целых решений.

При этом из $m > g$, $m > n - g$, получаем: $n - m < g < m$, из чего следует $n - m + 1 < g + 1 < m + 1$. Иначе говоря, $n - m + 1 = \min\{m + 1, g + 1, n - m + 1\}$.

Кроме того, из условия $m > g$ получаем: $-m < -g \Rightarrow n - m + 1 < n - g + 1$. Следовательно,

$$n - m + 1 = \min\{m + 1, g + 1, n - g + 1, n - m + 1\}.$$

Например, пусть $g = 4$, $m = 6$ и $n = 8$. Тогда формула (2) имеет вид

$$D_8^6 = D_4^2(X^1)D_4^4(X^2) \vee D_4^3(X^1)D_4^3(X^2) \vee D_4^4(X^1)D_4^2(X^2).$$

То есть количество слагаемых $sum = 3 = n - m + 1$.

Итак, во всех случаях получаем, что $sum = \min\{m + 1, g + 1, n - g + 1, n - m + 1\}$. Утверждение доказано.

Следствие 1. Количество слагаемых *sum* в формуле (1) зависит от m, $n - m$, k и $n - k$, а именно $sum = \min\{m + 1, k + 1, n - k + 1, n - m + 1\}$.

Доказательство. Так как формула (1) является частным случаем формулы (2), а именно получается из формулы (2) при $g = k$, то $c = \min\{m + 1, k + 1, n - k + 1, n - m + 1\}$. Следствие доказано.

Каждое из слагаемых текущего шага разложения реализуется некоторой подсхемой. Выясним, какая их них даст наибольшую длину пути. В силу того, что C_n^m достигает максимума при $m = n/2$, таким слагаемым будет слагаемое, у которого i наиболее близкое к $g/2$ или $m - i$, наиболее близкое к $(n - g)/2$.

Приведем алгоритм нахождения максимальной длины пути L в детекторе 1 для D_n^m, где $0° < m° \le n$, k – число входов в ПЛБ. Как было сказано выше, для нахождения максимальной длины пути в подсхеме на текущем шаге разложения следует выбирать из всех множителей такое D_p^q, у которого $p = \max(k, n - k)$, а q – число, наиболее близкое к $p/2$.

Найдем число μ шагов разложения. Так как количество подмножеств переменных мощности k (одно из подмножеств может иметь мощность меньше k), на которые разбивается множество переменных мощности n равно $\lceil n/k \rceil$, то число шагов разложения $\mu = \lceil n/k \rceil - 1$.

По следствию 1, число слагаемых *sum* в формуле разложения для различных D_p^q находится по формуле $sum = \min(k + 1, p - k + 1, q + 1, p - q + 1)$.

Теперь найдем *CountCLB* – количество ПЛБ, необходимых для реализации *sum* слагаемых. В каждом слагаемом два множителя и каждый множитель – это отдельный вход в некоторый ПЛБ. Так как число входов в ПЛБ равно k, то для реализации всех *sum* слагаемых потребуется $CountCLB = \lceil 2sum/k \rceil$ ПЛБ.

Так как $sum = \min(k + 1, p - k + 1, q + 1, p - q + 1)$, то $sum \le k+1$. Следовательно, $CountCLB = \lceil 2sum/k \rceil \le \lceil (2k + 2)/k \rceil \le 3$. Поэтому максимальная длина пути λ_{max} в подсхеме для реализации формулы разложения на каждом шаге не может быть больше 2. В итоге получаем, если $CountCLB \ge 2$, то реализация формулы потребует длины пути в подсхеме $\lambda = 2$, иначе $\lambda = 1$. Особая ситуация возникает на первом шаге разложения, т.е. при $\mu = 1$:

а) если $CountCLB = 2$, то выходы этих ПЛБ и есть выходы детектора, т.е. $\lambda = 1$;

б) если $CountCLB = 3$, то, как и в общем случае, $\lambda = 2$.

***Алгоритм нахождения L для детектора* 1**

1. Положим $q = m$, $p = n$.
2. Вычислим число шагов разложения $\mu = \lceil p/k \rceil - 1$, $L = 1$.
3. Вычислим число слагаемых $sum = \min(k + 1, p - k + 1, q + 1, p - q + 1)$.
4. Вычислим количество ПЛБ *CountCLB*, необходимых для реализации *sum* слагаемых $CountCLB = \lceil 2sum/k \rceil$.
5. Определяем λ:

а) если имеем первый уровень разложения (т.е. μ = 1) и количество *CountCLB*, необходимых для реализации формулы ПЛБ, равно 3, то λ = 2, в противном случае λ = 1;

б) если не первый уровень разложения (т.е. μ ≠ 1) и количество *CountCLB*, необходимых для реализации формулы ПЛБ, ≥ 2, то λ = 2, в противном случае λ = 1.

6. $L = L + \lambda$, $\mu = \mu - 1$.
7. Если μ = 0, то нахождение *L* закончено. Иначе, полагаем $p = \max(p, p - k)$, *q* – число, наиболее близкое к *p*/2, и переходим на шаг 3.

Приведем алгоритм нахождения максимальной длины пути *L* в детекторе 2 для D_n^m, где 0°<°*m*°<°*n*, *k* – число входов в ПЛБ.

В детекторе 2 наибольшее число слагаемых $sum_{\max}$ на каждом шаге разложения может сильно отличаться от $sum_{\max}$ предыдущего шага разложения (наибольшее число на первом шаге разложения; на каждом последующем шаге число слагаемых не изменяется или становится меньше). Это обусловлено тем, что для данного *n* C_n^m достигает максимального значения при $m = n/2$ и с уменьшением *n* C_n^m уменьшается. Увеличение числа слагаемых в формуле разложения может увеличить число ПЛБ в детекторе, что способно привести к увеличению максимальной длины пути в детекторе.

Разложение по формуле (2) заканчивается, если мощность каждого из подмножеств переменных становится меньше либо равной *k* (последний шаг разложения). По утверждению 1, число слагаемых *sum* в формуле разложения для различных D_n^m находится по формуле $sum = \min\{m + 1, g° + 1, n - g + 1, n - m + 1\}$. Теперь найдем *CountCLB* – количество ПЛБ, необходимых для реализации *sum* слагаемых. В каждом слагаемом два множителя и каждый множитель – это отдельный вход в некоторый ПЛБ. Так как число входов в ПЛБ равно *k*, то для реализации всех *x* слагаемых потребуется $CountCLB = \lceil 2sum/k \rceil$ ПЛБ. Но если *CountCLB* > 1 (не первый шаг разложения) или если *CountCLB*°>°2 (первый шаг разложения), то требуются дополнительные ПЛБ для реализации полученного числа *CountCLB* ПЛБ. Это увеличивает максимальную длину пути $\lambda_{\max}$ в подсхеме, реализующей формулу разложения. Максимальная длина пути $\lambda_{\max}$ увеличивается до тех пор, пока количество дополнительных ПЛБ, необходимых для реализации вновь полученного числа *CountCLB* ПЛБ, не станет равным 1 (2).

***Алгоритм нахождения L для детектора* 2**

1. Положим $q = m$, $p = n$.
2. $L = 1$.
3. Найдем такие *g* и *p* – *g*, что $p = g + (p - g) = kq^1 + kq^2 + r$, где *k* – число входов в ПЛБ, $r < k$, а kq^1 – ближайшее к $\lceil p/2 \rceil$ число, кратное *k*.
4. Вычислим $sum = \min(g + 1, p - g + 1, q + 1, p - q + 1)$ и $CountCLB = \lceil 2sum/k \rceil$, λ = 1.
5. Если не первый шаг разложения и *CountCLB* > 1, то до тех пор, пока количество необходимых для реализации формулы ПЛБ $CountCLB = \lceil CountCLB/k \rceil$ больше 1, увеличиваем λ.

6. Если первый шаг разложения и $CountCLB > 2$, то до тех пор, пока количество необходимых для реализации формулы ПЛБ $CountCLB = \lceil CountCLB / k \rceil$ больше 2, увеличиваем λ.

7. $L = L + \lambda$.

8. Если $g \le k$ и $p - g \le k$, то нахождение L закончено. Иначе $p = g$, q – число, наиболее близкое к $p/2$ и идем на шаг 3.

Сравним сложность детектора 1 и детектора 2. Будем считать, что при четных k используются двухвыходные ПЛБ (табл. 2).

Таблица 2

Сравнение сложности детектора 1 и детектора 2

m	n	k	Оценка максимальной длины пути		Количество ПЛБ	
			Детектор 1	Детектор 2	Детектор 1	Детектор 2
6	12	4	5	5	31	31
8	16	4	7	5	55	60
3	16	4	6	4	32	32
10	20	4	9	7	85	82
3	20	4	8	6	43	43
3	64	4	30	8	164	164
10	64	4	31	11	448	471
20	64	4	31	12	703	728
32	64	4	31	12	811	840

Количество ПЛБ, сопоставляемых листьям дерева разложения (все ПЛБ реализуют функции вида D_k^q, где $0 \le q \le k$) одинаково при любом из рассматриваемых способах разбиения (так как в обоих случаях при разбиении идет ориентация на число k входов в ПЛБ). Разница в количестве ПЛБ начинается при многократном применении формул (1) и (2) при реализации D_p^q, где $p > k$.

Заключение

Таким образом, при $n > 3k$ оценки максимальных длин путей детекторов, синтезированных методом, предлагаемым в [9], оказываются меньше, чем оценки длин путей детекторов, синтезированных методом, предлагаемым в [8]. Количество ПЛБ, используемых при синтезе детектора рассматриваемыми методами, существенно не отличается.

ЛИТЕРАТУРА

1. Parag K.L. Self-Checking and Fault-Tolerant Digital Design. Morgan Kaufmann Pub., 2000. 400 p.
2. Anderson D.A., Metze G. Design of totally self-checking check circuits for m-out-of-n codes // IEEE Trans. Computers C-22. 1973. P. 263–269.
3. Marouf M.A., Friedman A.D. Efficient design of sel-checking checker for any m-out-of-n code // IEEE Trans. Computers C-27. 1978. P. 482–90.
4. Ubar R., Raik J., Vierhaus H.-T. Design and Test Technology for Dependable Systems-on-Chip. New York : IGI Global, 2011. 578 p.
5. Blanke M., Kinnaert M., Lunze J., Staroswiecki M. Diagnosis and Fault-Tolerant Control. 2nd ed. Berlin : Springer-Verlag Berlin Heidelberg, 2006. 672 p.
6. Fujiwara E. Code Design for Dependable Systems: Theory and Practical Applications. John Wiley & Sons, Inc., 2006. 720 p.
7. Göessel M., Ocheretny V., Sogomonyan E., Marienfeld D. New Methods of Concurrent Checking. Ed. 1. Dordrecht : Springer Science + Business Media B. V., 2008. 182 p.
8. Матросова А.Ю., Никитин К.В. Синтез самотестируемого детектора (m, n)-кодов на программируемых логических блоках // Вестник Томского государственного университета. Приложение. 2003. № 6. С. 124–136.
9. Буторина Н.Б., Цидендоржиева С.Р. Построение самотестируемого детектора равновесных кодовых слов // Новые информационные технологии в исследовании сложных структур : 7-я рос. конф. с междунар. участием. Томск : Том. гос. ун-т, 2008. С. 44.

Поступила в редакцию 7 сентября 2018 г.

Butorina N.B., Pakhomova E.G. (2019) ON THE LENGTHS OF THE PATHS IN SELF-TESTING CHECKERS BASED ON CLB. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 48. pp. 92–102

DOI: 10.17223/19988605/48/11

Self-testing checkers, in particular, (*m*, *n*)-code checkers are widely used in self-checking circuits design. There is a special formula $D_n^m(X) = \sum_{i=0}^{m} D_g^i(X^1) D_{n-g}^{m-i}(X^2)$ representing all (*m*, *n*)-code words. This paper compares two self-testing (*m*, *n*)-code checkers designed on the base of configurable logical blocks (CLB) by using a multiple decomposition of the formula. The form of the formula obtained by decomposition depends on the set of represented code words (i.e., values of *m* and *n*), the number *k* of the CLB inputs and the way of partitioning variables into subsets. In the first approach, $g = k$. If $n - g > k$ then the decomposition formula is applied again for the obtained function $D_{n-g}^{m-i}(X^2)$, etc. In the second approach, *g* is the least integer which is greater than or equal to *n*/2. If $g > k$ and/or $n - g > k$, then the decomposition formula is applied again for all obtained functions $D_g^i(X^1)$, $D_{n-g}^{m-i}(X^2)$, $i = \overline{0,m}$, etc. In the paper, we analyze how the choice of g (or the way of partitioning variables into subsets) influence on the maximum path length in the designed checker and on the number of CLBs in the checker.

Keywords: selftesting; checker; configurable logical blocks (CLB); (*m*, *n*)-code.

BUTORINA Nataly Borisovna (National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: nnatta07@mail.ru

PAKHOMOVA Elena Grigorievna **(**Candidate of Physics and Mathematics, Associate Professor, National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: peg@tpu.ru

REFERENCES

1. Parag, K.L. (2000) *Self-Checking and Fault-Tolerant Digital Design*. Morgan Kaufmann.
2. Anderson, D.A. & Metze, G. (1973) Design of totally self-checking check circuits for m-out-of-n codes. *IEEE Trans. Computers* C-22. pp. 263–69. DOI: 10.1109/T-C.1973.223705
3. Marouf, M.A. & Friedman, A.D. (1978) Efficient design of sel-checking checker for any m-out-of-n code. *IEEE Trans. Computers* C-27. pp. 482–90. DOI: 10.1109/TC.1978.1675138
4. Ubar, R., Raik, J. & Vierhaus, H-T. (2011) *Design and Test Technology for Dependable Systems-on-Chip*. New York: IGI Global.
5. Blanke, M., Kinnaert, M., Lunze, J. & Staroswiecki, M. (2006) *Diagnosis and Fault-Tolerant Control*. 2nd ed. Berlin: Springer-Verlag Berlin Heidelberg.
6. Fujiwara, E. (2006) *Code Design for Dependable Systems*: *Theory and Practical Applications.* John Wiley & Sons.
7. Göessel, M., Ocheretny, V., Sogomonyan, E. & Marienfeld, D. (2008) *New Methods of Concurrent Checking*. Dordrecht: Springer Science+Business Media B. V.
8. Matrosova, A.Yu. & Nikitin, K.V. (2003) Sintez samotestiruemogo detektora (m, n)-kodov na programmiruemykh logicheskikh blokakh [Design of a self-testing (m,n)-code checkers using configurable logic blocks]. *Vestnik Tomskogo gosudarstvennogo universiteta. Prilozhenie*. 6. pp. 124–136.
9. Butorina, N.B. & Tsidendorzhieva, S.R. (2008) [Design of a self-testing (m,n)-code checkers]. *Novye informatsionnye tekhnologii v issledovanii slozhnykh struktur* [Computer-aided technologies in applied mathematics. ICAM 2008]. Proc. of the 7th Conference. Tomsk: Tomsk State University. pp. 44. (In Russian).