УДК 681.3.001.2
DOI 10.18413/2687-0932-2020-47-3-583-599

Метод синтеза формирователя тестовой последовательности с перестраиваемыми параметрами, основанный на представлении логических функций в обобщенной форме

В.Г. Рубанов, Е.Н. Коробкова, О.В. Луценко
Белгородский государственный технологический университет им. В.Г. Шухова,
Россия, 308012, г. Белгород, ул. Костюкова, 46
E-mail: tk.bstu@gmail.com

Аннотация
Предложен метод синтеза формирователя тестовых сигналов с перестраиваемыми временными параметрами тестового контента в зависимости от необходимости изменения глубины контроля в процессе оценки технического состояния синтезируемого цифрового автомата. Используется представление логических функций в обобщенной форме, когда рассматривается более широкая трактовка основной теоремы алгебры логики, предполагающая введение дополнительных переменных, кроме нуля и единицы, которые могу приобретать значения функции в точках её области определения. Излагается методика синтеза структуры цифрового автомата, генерирующего тестовую периодическую посылку, заданной конфигурации. Иллюстрация методики проведена на версии алгоритма нахождения кодов настройки на заданный режим, пригодной для программной реализации. Приведен пример, демонстрирующий реализацию заданного теста с произвольной конфигурацией.

Ключевые слова: работоспособность, контроль, диагностика, тестовый контент, формирователь, цифровой автомат, настроечные переменные, обобщенная логическая функция, минтермы, литералы, импликанты.

Благодарности: работа выполнена при финансовой поддержке Министерства науки и высшего образования Российской Федерации в рамках соглашения № 075-11-2019-070 от 29.11.2019 г.

Для цитирования: Рубанов В.Г., Коробкова Е.Н., Луценко О.В. 2020. Метод синтеза формирователя тестовой последовательности с перестраиваемыми параметрами, основанный на представлении логических функций в обобщенной форме. Экономика. Информатика. 47 (3): 583–599. DOI 10.18413/2687-0932-2020-47-3-583-599.

Synthesis method for a test sequence generator with tunable parameters, based on the presentation of logical function in a generalized form

V.G. Rubanov, E.N. Korobkova, O.V. Lutsenko
Belgorod State Technological University named after V.G. Shukhov,
Russia, 46 st. Kostyukov, Belgorod, 308012, Russia
E-mail: tk.bstu@gmail.com

Abstract
Methods for the synthesis of test sequence generators as digital automata with restructured test content parameters are discussed in this article. Parameters of test content are restructured depending on the need to expand it in the process of monitoring the technical condition of a digital device. The synthesis technique of the structure of a digital automaton is described. This automaton generates a test set of a given configuration based on the presentation of logical functions in a generalized form. The analysis of restructuring options oriented to the use of a totalizing counter is carried out. Three versions of the algorithm for finding tuning codes for specified modes are considered. The complexity of the algorithms for finding tuning codes is estimated. It is shown that for all their simplicity they are cumbersome and therefore it is advisable to obtain

an algorithm that is executed in software.The illustration of the technique is carried out on one of the versions of the algorithm suitable for software implementation. An example is presented that demonstrates the implementation of the given tests with an arbitrary configuration.

Keywords: working capacity, control, diagnostics, test content, shaper, digital machine, tuning variables, generalized logic function, minterm, literal, implicants.

Acknowledgements: the work is executed at support of the grant: the Public contract project 03/19 from 03.09.2019 in the framework of agreement №075-11-2019-070 of 29.11.2019 (unique number 07519SU2000000).

For citation: Rubanov V.G., Korobkova E.N., Lutsenko O.V. 2020. Synthesis method for a test sequence generator with tunable parameters, based on the presentation of logical function in a generalized form. Economics. Information technologies. 47 (3): 583–599 (in Russian). DOI 10.18413/2687-0932-2020-47-3-583-599.

Введение

Контроль работоспособности, т. е. выявление ошибок, и диагностика неисправных цепей СБИС являются важным этапом в производстве микросхем с точки зрения обеспечения их надежности. Неразрушающий контроль предполагает использование тестовых сигналов, обнаруживающих неисправности, причем в процессе испытаний часто необходимо создавать дополнительные тестовые наборы [Sarmad et al., 2016]. Известно достаточно много методов не только обнаружения ошибок, но и диагностики неисправностей [Desineni et al., 2006; Lin et al., 2008; Wang, Wei, 2009; Yu, Blanton, 2012; Xue et al., 2013; Riefert et al., 2015], в которых используют генерацию тестовых шаблонов, применяемых к чипам, а затем с помощью контролирующего оборудования получают отклики, представляющие журнал ошибок (FL). Если для пары неисправностей не существует отличительного теста, то они называются неразличимыми [Amyeen et al., 2016]. Для обнаружения функционально-эквивалентных неисправностей, т. е. неразличимых пар неисправностей, в работах [Pomeranz, 2012; Li et al., 2015] было предложено вводить дополнительные контрольные точки в схеме. Авторы статьи [Wu et al., 2018] предлагают идентифицировать характер и место функционально-эквивалентной неисправности за счет использования избыточных элементов в схеме. Кроме того, в этой статье отмечается, что применяемые дополнительные тесты с логикой восстановления позволяют различать ошибки в подмножествах из более чем двух ошибок, что улучшает диагностическое разрешение.

В [Medina et al., 1990; Mahlstedt, Koopmeiners, 1991; Bartenstein, 2000] ATPG-инструмент приспособлен для генерации отличительного теста, а в [Bhatti, Blanton, 2006] обоснованы условия логической выполняемости (SAT) для различения произвольных ошибок. Наряду с указанными подходами для генерации тестов в последнее время предлагаются методы, основанные на обучении [Chandrasekar, Hsiao, 2009; Chandrasekar et al., 2010;], применении генетических алгоритмов [Corno et al., 1995; Girard et al., 1996; Bernardi, et al., 2008]. Для производственных условий интерес представляет новая методология, базирующаяся на последовательном выполнении процедуры диагноза [Amyeen et al., 2016], когда первичный тестовый набор предназначен для покрытия дефектов, не обеспечивая максимального диагностического решения, т. е. сначала генерируется детерминистическое содержание тестов, чтобы отличить подозреваемые узлы или линии, а затем генерируются дополнительные ориентированные тесты для дополнения контента.

Повышение сложности аппаратуры цифровых систем управления приводит к увеличению вероятности появления неисправностей в ней, что несомненно ведёт к дополнительным материальным затратам, предотвратить которые можно только путем использования известных контроля и диагностики [Wang, Wei, 2009] или разработка новых способов применения тестопригодного обеспечения, создаваемого на стадии проектирования систем и последующего их сервисного обслуживания при производстве и эксплуатации. Решение задач проектирования контрольно-диагностических тестов, алгоритмов контроля и

поиска дефектов неразрывно связано с вопросами моделирования контролируемых цифровых систем на функциональном, алгоритмическом и структурном уровнях.

Как видно из анализа работ, ряд методов контроля работоспособности и поиска неисправностей используют синхронные модели неисправного поведения, на чем строятся алгоритмы одиночного, параллельного, дедуктивного, совместного и кубического моделирования [Бондаренко и др., 2000], при этом последний из них считается наиболее эффективным. Его применение предполагает наличие табличной формы описания функциональных блоков или элементов с последующим определением частных, кратных или векторных булевых производных, которые характеризуют изменение значений функции $F(x)$ при изменении её аргументов по аналогии с классическим дифференциальным исчислением с одним существенным отличием, когда булева производная по переменной x_i не зависит от этой переменной, а классическая частная производная в общем случае зависит от неё [Рубанов, Коробкова, 2008]

Таким образом, анализируя различные подходы к осуществлению процедур контроля работоспособности и диагностики цифровых устройств, можно заметить, что в любом случае первичным устройством при генерировании тестирующих сигналов является формирователь тестов, причем последний должен обладать свойством широкой перестройки параметров тестовой посылки. Это свойство позволит осуществлять гибкую перестройку, расширяя при необходимости тестовый контент.

В настоящей работе предлагается подход к формированию тестового контента с гибкой перестройкой в соответствии с изменяемыми параметрами тестирующей последовательности.

Этот способ основан на синтезе структуры формирователя, использующем представление логических функций в обобщенной форме [Рубанов, Коробкова, 2008; Рубанов, Коробкова, 2009], когда трактовка основной теоремы алгебры логики (полного разложения Шеннона) трактуется в предположении, что функция принимает значение нуля или единицы в точках ее области определения, но также её значения могут быть и другими, например, $a,b.c,...$ или функциям от них. Вводимое понятие обобщенной логической функции (ОЛФ) не является исключительно новым, поскольку к этому классу функций относятся функции алгебры логики, зависящие от логических переменных и принимающие значения переменных из того же множества, что и множество аргументов этих функций. Особенность предлагаемого понятия ОЛФ состоит только в специфике представления их и последующего преобразования, обеспечивающих возможность упрощения некоторых вопросов, связанных с анализом и синтезом цифровых устройств.

Каноническими представлениями традиционных логических функций являются совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ). Такие же понятия вводятся для обобщенных логических функций [Рубанов, Коробкова, 2008]. Отличие состоит в том, что в этом случае СДНФ в наиболее общем виде полного разложения Шеннона значения функции $f_i = \{0,1,a,b,...\}$ будет содержать параметры $(a,b,...)$, являющиеся в свою очередь некоторыми зависимостями от других переменных.

Следует подчеркнуть, что в эту статью мы не включаем вопросы синтеза вида, т. е. конфигураций, диагностического теста для того или иного цифрового устройства, а ограничиваемся только способом реализации заданного теста аналитическими методами, позволяющими получать структуру формирователя.

Постановка задачи

Для типовых узлов устройств управления, построеныых на микроконтроллерах, существуют тесты достаточной полноты, это позволяет в качестве исходной посылки считать, что форма тестового сигнала известна [Чжен и др., 1972; Бессонов и др., 1986; Горовой и др., 1990].

Целью работы является синтез структуры перестраиваемого формирователя тестов на основе обобщенных логических функций (ОЛФ), обеспечивающего обнаружение неисправности цифрового устройства при проведении контроля работоспособности и диагностики.

Анализ различных видов тестовых сигналов позволил в общем случае представить их в форме некоторой функции вида $F(\varepsilon,\tau,T,t)$, обладающей свойством периодичности с периодом T, уровнем импульсов 0 или 1 и изменяющейся задержкой ε, зависящей от контролируемой схемы. Структура синтезируемого формирователя должна обеспечивать изменение параметров функции F в процессе контроля, т. е. адаптацию к контролируемому устройству поэтапно в соответствие с процедурой контроля, которой и определяется тестовый контент на каждом этапе контроля.

Решение задачи на основе использования обобщенных логических функций

Вспомогательный сигнал, подаваемый на вход формирователя, представляет собой периодическую последовательность тактирующих импульсов, определяемую параметрами: T_o – период, τ – длительность импульса. Если длительность импульса равна длительности паузы, т. е. $\tau = T_o/2$, то такую последовательность называют меандр. На выходе формирователя, представляющего собой цифровой автомат с перестраиваемыми параметрами тестовой последовательности, формируется также периодическая последовательность дискретных интервалов времени с периодом $T = kT_o$, где k – целое число. Тогда выходной сигнал формирователя можно представить в виде периодической функции времени как

$$F(\varepsilon,\tau,T,t) = F^{T}_{\varepsilon,\tau}(t), \quad (1)$$

где ε – величина задержки, т. е. отрезок времени между началом отсчета и моментом появления первого импульса.

Решение задачи синтеза цифрового автомата (ЦА) с перестраиваемой длительностью выходных параметров периодической последовательности можно осуществить как синтез автомата Мили, который может представлять собой известную схему на суммирующем или вычитающем счетчике с настройкой последнего на заданное число состояний, определяющее длину цикла проектируемого цифрового автомата с перестраиваемыми параметрами (ЦА ПП), кроме того можно использовать сдвигающие регистры с кольцевой обратной связью.

В связи с этим задача проектирования сводится к нахождению некоторых L последовательно идущих состояний из имеющихся k $(L < k)$ для циклического устройства, которые бы обеспечивали формирование выходного сигнала и выбор значений кодов настройки.

Каждому из R вариантов настройки могут соответствовать любые L последовательно идущих состояний счетчика из k возможных. Тогда в каждом из R вариантов можно выбрать k подвариантов, отличающихся параметром периодической функции, например, началом и окончанием формирования выходного импульса в одном цикле, т. е. привязкой и размещением формируемого выходного импульса к конкретным состояниям циклического устройства. Даже при небольшом диапазоне перестройки формирователя тестов число всех возможных вариантов привязки, равное k^R, довольно велико. Кроме того, существует соответствие между вариантами настройки и наборами настроечных переменных, образующих минтермы. Число таких способов определяется числом перестановок.

Предлагается способ проектирования ЦА с перестраиваемой (программируемой) длительностью дискретных интервалов, обеспечивающий выбор оптимального варианта схемной реализации, базирующийся на представлении функции выхода в форме ОЛФ с зависимыми параметрами, значение первичных переменных этой функции определяется

состоянием триггеров, входящих в циклическое устройство, а параметрами её будут минтермы, определяющие какой-то из вариантов настройки.

Тогда первичные переменные, с одной стороны, можно трактовать как такты на оси автоматного времени, а с другой – как состояние циклического устройства (рис. 1). В качестве циклического устройства будем использовать, как отмечено выше, двоичные суммирующие или вычитающие счетчики.

Таким образом, функция выхода определяется взаимным положением виртуальных интервалов (импульсов) на оси автоматного времени, причем состояние счетчика связано с тактами переключений, а значение функции выхода в этих тактах отвечает логической сумме минтермов. Каждый вариант настройки определяется соответствующим минтермом.

Если в качестве циклического устройства используется двоичный суммирующий счетчик, а шаг перестройки длительности выходных интервалов во всем диапазоне принимается равным единице, то первый такт (начало формирования) интервалов четной кратности для всех вариантов настройки «привязывается» к одному и тому же четному состоянию счетчика – i, а в первый такт формирования интервалов нечетной кратности – к следующему (соседнему) нечетному состоянию суммирующего счётчика $(i+1)$. При использовании вычитающих счетчиков первый такт формирования интервалов четной кратности для всех вариантов настройки «привязывается» к одному и тому же нечетному состоянию счетчика – j, а первый такт формирования интервалов нечетной кратности к следующему (соседнему) четному состоянию вычитающего счетчика $(j-1)$.

Проиллюстрируем общие принципы оптимального построения для одного диапазона перестройки, которые можно распространить на любой другой диапазон.

На рис. 1 приведены виртуальные эпюры ЦА ПП. Здесь длина цикла является фиксированной и равна $k=16$, шаг перестройки равен T, а диапазон перестройки лежит в пределах от T до $8T$.

Варианты настройки представлены каждый на отдельной оси автоматного времени виртуальными импульсами, обозначенными минтермами, образуемыми литералами настроечных переменных, определяющих режимы (варианты) настройки ЦА ПП. Ось автоматного времени для случая использования суммирующего счетчика $(i, i+1, i+2, ..., i+15)$ – обозначение сверху, где i – четное, соответственно разметка оси для вычитающего счетчика $(j, j-1, ...)$ – снизу.

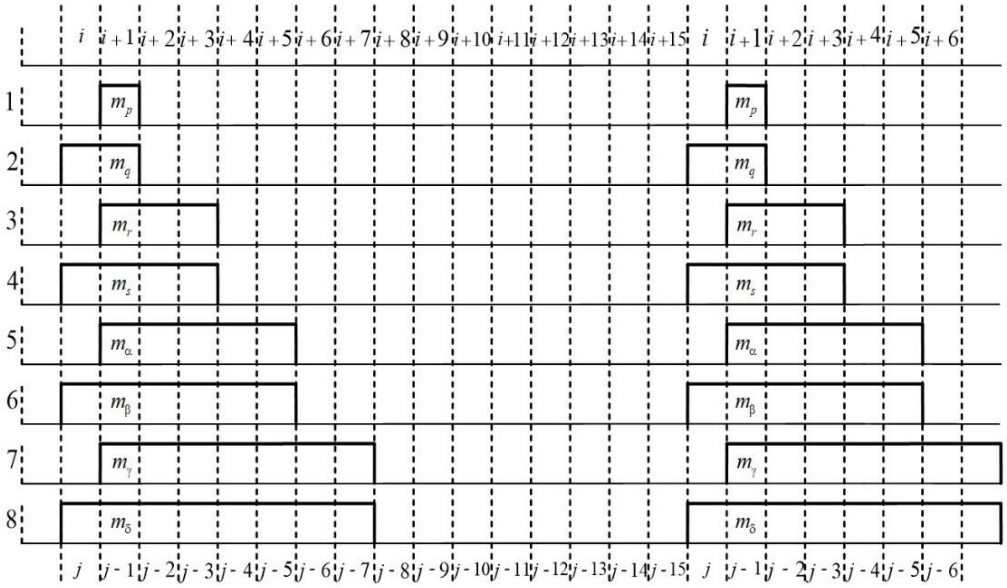

Рис. 1. Виртуальные эпюры способа размещения
Fig. 1. Virtual plots of the placement method

Минтерм m_p определяет (настраивает) формирование временных интервалов длительностью, равной длительности одного такта автоматного времени $(T): m_p - 2T; m_r - 3T; m_s - 4T; m_\alpha - 5T; m_\beta - 6T; m_\gamma - 7T; m_\delta - 8T$.

Возможен второй способ привязки, эпюры которого представляют собой зеркальное отображение первого.

Изучение виртуальных эпюр, соответствующих конкретным диапазонам перестройки, позволило установить некоторую общность, позволяющую распространить алгоритм привязки и размещения на самый общий случай с произвольно принятой размерностью диапазона перестройки с количеством настроечных переменных r. Так, если шаг перестройки совпадает с длительностью одного такта, то: количество минтермов и тактов равны и совпадают с количеством вариантов перестройки, равным 2^r; число минтермов, определяющих функцию в i-м такте, меньше на 2, а в $(i+1)$-м такте оно также равно числу состояний счетчика, т. е. 2^r; в каждой последующей паре тактов число минтермов уменьшается на два по отношению к предыдущей паре, тогда число минтермов, которые определяют функцию в последних двух тактах диапазона, равно двум.

Обозначим пары тактов i-й, и $(i+1)$-й как нулевую пару $p_0 = 0$; $(i+2)$-й и $(i+3)$-й такты – как первую пару – $p_1 = 0$; и т. д., и получим общую формулу, устанавливающую связь между числом минтермов N в каждой такой паре, числом настроечных переменных и номером пары:

$$N = 2^r - 2^P.$$

Представим теперь режимы настройки в картах с соседним кодированием. Предложенный подход к выделению общего массива состояний счетчика для всех вариантов настройки резко сокращает суммарное число импликант, покрывающих все множество единичных значений функции выхода в целом.

Поскольку каждый такт автоматного времени и отвечающее ему состояние счетчика можно обозначить в виде некоторой точки области определения функции выхода, зависящей от состояния счетчика, а значение функции выхода ЦА ПП в этом такте, определяемое логической суммой минтермов, можно рассматривать как значение функции в этой точке, то функцию выхода в целом можно представить в форме функции с зависимыми параметрами, в частности в СДНФ:

$$F = \vee f_i \times K_i, \tag{2}$$

где K_i – минтермы, определяющие состояние счетчика, образуемые литералами выходных сигналов счетчика $\tilde{Q}_3\tilde{Q}_2\tilde{Q}_1\tilde{Q}_0$; f_i – значение функции на выходе, соответствующее i-му состоянию счетчика, определяемое логической сумой минтермов, образуемых настроечными переменными $(m_i = \tilde{a}_3\tilde{a}_2\tilde{a}_1\tilde{a}_{0i})$.

Для ЦА ПП с виртуальными эпюрами, показанными на рис. 1, значения функций в точках области определения находится как:

$$f_i = m_q \vee m_s \vee m_\beta \vee m_\delta;$$
$$f_{i+1} = m_p \vee m_q \vee m_r \vee m_s \vee m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta = 1;$$
$$f_{i+2} = m_r \vee m_s \vee m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta;$$
$$f_{i+3} = m_r \vee m_s \vee m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta;$$
$$f_{i+4} = m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta;$$
$$f_{i+5} = m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta;$$
$$f_{i+6} = m_\gamma \vee m_\delta;$$
$$f_{i+7} = m_\gamma \vee m_\delta.$$

В практие проектирования более рационально представлять функцию выхода в форме минимальной ДНФ, сложность которой зависит от привязки формируемых интервалов к состояниям циклического устройства. Возникает проблема нахождения оптимального варианта привязки, обеспечивающего минимально возможный вариант минимальной ДНФ.

Анализ размещения каждого варианта настройки предлагаем проводить в карте с соседним кодированием, где координаты клеток отождествляются с состояниями счетчика, а значения функции в них представлены логической сумой минтермов (f_i) или нулем в зависимости от значений, имеющих место в соответствующих тактах оси автоматного времени.

Расстановку ненулевых значений функции (логических сумм минтермов) можно начинать с любых четных номеров клеток (для схемы, выполненной на суммирующем счетчике). Это даёт возможность выбрать такое положение начальной точки формирования временных интервалов для всех вариантов настройки, которому будет соответствовать массив клеток, содержащих логические суммы минтермов, покрываемый минимально возможным числом правильных конфигураций максимально возможной площади. Следовательно, возникает возможность формирования массива, путем выполнения требования минимально возможного покрытия выделенных групп логических сумм минтермов с последующим оптимальным их кодированием.

Под оптимальным подразумевается такое кодирование, при котором логической сумме некоторой группы минтермов, образованных литералами настроечных переменных, будет соответствовать тот или другой литерал одной из них. Если число настроечных переменных равно r, то, как было отмечено выше, общее число минтермов равно 2^r. Множество этих минтермов M можно представить в виде r толерантных подмножеств: $P_1 \subset M, P_2 \subset M, ..., P_r \subset M.$

Отношение толерантности представляет собой экспликацию интуитивных представлений о сходстве и неразличимости. В нашем случае каждое подмножество неразличимо само с собой, а сходство двух подмножеств не зависит от того, в каком порядке они рассматриваются.

Отношение толерантности на множестве подмножеств P должно удовлетворять не только рефлексивности и симметричности, но и транзитивности, т. е. все подмножества сходны между собой. Толерантность задаем тремя основными отношениями (условиями): каждое подмножество содержит 2^{r-1} минтермов; два любых произвольно выбранных подмножества из множества P содержат 2^{r-2} общих минтерма; минтермы, входящие в каждое подмножество, взаимно-соседние.

Таким образом, справедливо утверждение, что каждому из r подмножеств существует его дополнение, имеющее 2^{r-1} минтермов из множества M, которые не вошли в исходное подмножество:

$$\overline{P}_1 = M \setminus P_1;\ \overline{P}_2 = M \setminus P_2;\ ...;\ \overline{P}_r = M \setminus P_r\,. \qquad (3)$$

В результате получаем еще r подмножеств, определенных теми же самыми отношениями толерантности. Более того, каждое из прямых исходных подмножеств толерантно к любому инверсному подмножеству кроме собственного дополнения и наоборот. В соответствии с приведенными выше условиями толерантности можно утверждать: логической сумме минтермов, образующих каждое из подмножеств, отвечает литерал, отличный от других; логическим суммам минтермов, входящих в подмножество P_i и инверсное ему подмножество $\overline{P}_i$ отвечает литерал $\tilde{a}_k$, и его инверсное значение соответственно.

Эти положения позволяют формировать группы минтермов, образующих правильные конфигурации максимально возможной площади, которым соответствуют импликанты минимально возможного ранга, и осуществлять оптимальное кодирование этих групп.

Анализ всевозможных вариантов представления заданного диапазона перестройки в картах с последующим нахождением минимальной ДНФ функции выхода для разных вариантов привязки начала формирования посылки осуществим на примере синтеза ЦА ПП с фиксированной длительностью цикла $k=16$, шагом перестройки длительности интервалов, равным периоду следования входного меандра T, в диапазоне перестройки от одного до восьми тактов $(T-8T)$.

Будем ориентироваться на применение в качестве базового элемента – суммирующего счетчика. Привязка осуществляется к одному и тому же четному состоянию счетчика (i) для интервалов четной кратности, а нечетной кратности – к состоянию $(i+1)$. Отсюда следует, что число вариантов привязки равно восьми, а именно $i=0,2,4,6,8,10,12,14$. На рис. 2 приведены варианты размещения с привязкой интервалов чётной кратности к нулевому состоянию счетчика, а нечётной к единичному. Как видно, функция выхода будет минимальна, если при кодировании минтермов, когда логическая сумма четырех минтермов, образованных литералами настроечных переменных $\tilde{a}_0, \tilde{a}_1, \tilde{a}_2$, будет представлять собой тот или другой литерал одной из настроечных переменных, т. е. будут выполнены условия смежности в группах: $m_q, m_s, m_\beta, m_\delta$; $m_r, m_s, m_\gamma, m_\delta$; $m_\alpha, m_\beta, m_\gamma, m_\delta$; m_γ, m_δ.

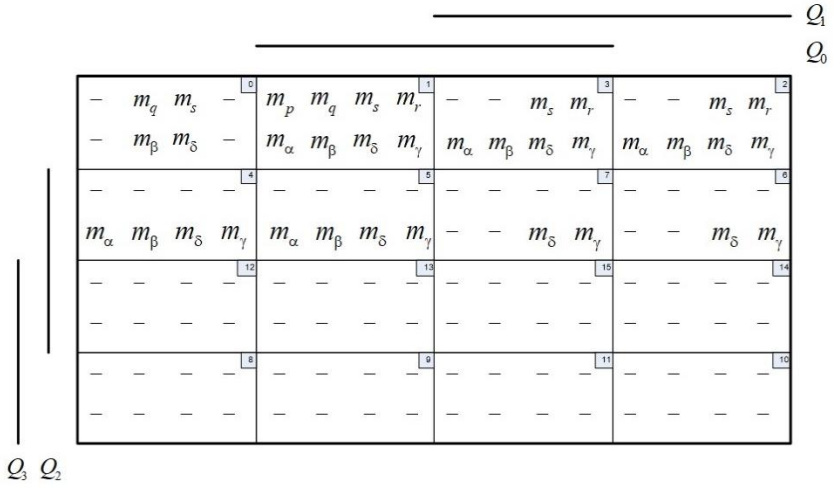

Рис. 2. Вариант рассматриваемой привязки и размещения
Fig. 2. Variant of the considered binding and placement

$$\begin{aligned} m_q \vee m_s \vee m_\beta \vee m_\delta &= \tilde{a}_i; \\ m_r \vee m_s \vee m_\gamma \vee m_\delta &= \tilde{a}_k; \\ m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta &= \tilde{a}_j. \end{aligned} \qquad (4)$$

Тогда логическая сумма двух соседних минтермов $m_\gamma \vee m_\delta$ представима в виде произведения второго ранга, образованного литералами $\tilde{a}_k$ и $\tilde{a}_j$, что при подстановке соответствующих логических сумм из (4) приведет к выражению: $\tilde{a}_j \times \tilde{a}_k = (m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta) \times (m_r \vee m_s \vee m_\gamma \vee m_\delta)$. Так как произведение разнотипных минтермов $m_a \times m_b$ при $(a \neq b)$, образуемых литералами одних и тех же переменных, тождественно равно нулю, то при выполнении операции перемножения произведения всех сомножителей, кроме $m_\gamma \times m_\gamma = m_\gamma$ и $m_\delta \times m_\delta = m_\delta$, будут равны нулю, что дает окончательный результат $\tilde{a}_j \times \tilde{a}_k = m_\delta \vee m_\gamma$.

Учет условий смежности позволяет каждую логическую сумму четырёх минтермов, отвечающих правильным конфигурациям, образуемых соседними клетками карты, покрыть

как один независимый параметр $(\tilde{a}_i, \tilde{a}_k, \tilde{a}_j)$, а логическую сумму $m_\delta \vee m_\gamma$ – как произведение независимых параметров $\tilde{a}_j \times \tilde{a}_k$.

Заменив логические суммы выделенных групп минтермов литералами соответствующих переменных и произведением литералов $\tilde{a}_j \times \tilde{a}_k$, можно упростить процесс выделения правильных конфигураций путем преобразования карты, изображенной на рис. 3.

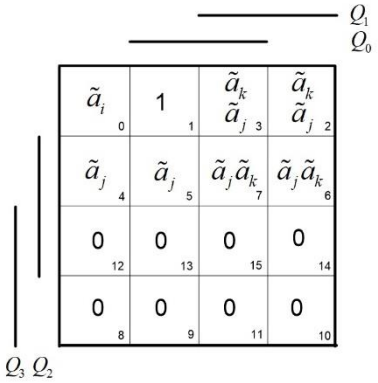

Рис. 3. Преобразованная карта первого варианта привязки и размещения
Fig. 3. The converted map of the first anchor and placement option

Представим номера клеток с правильными конфигурациями и соответствующие им простые импликанты для рассматриваемого варианта привязки (рис. 2) в форме:

$$\begin{cases} <1> - \overline{Q}_3\overline{Q}_2\overline{Q}_1Q_0; \\ <0,1> - (m_q \vee m_s \vee m_\beta \vee m_\delta)\overline{Q}_3\overline{Q}_2\overline{Q}_1 = \tilde{a}_i\overline{Q}_3\overline{Q}_2\overline{Q}_1; \\ <2,3> - (m_r \vee m_s \vee m_\gamma \vee m_\delta)\overline{Q}_3\overline{Q}_2Q_1 = \tilde{a}_k\overline{Q}_3\overline{Q}_2Q_1; \\ <2,3> - (m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta)\overline{Q}_3\overline{Q}_2Q_1 = \tilde{a}_j\overline{Q}_3\overline{Q}_2Q_1; \\ <4,5> - (m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta)\overline{Q}_3Q_2\overline{Q}_1 = \tilde{a}_j\overline{Q}_3Q_2\overline{Q}; \\ <2,3,6,7> - (m_\gamma \vee m_\delta)\overline{Q}_3Q_1 = \tilde{a}_j \times \tilde{a}_k\overline{Q}_3Q_1 \end{cases} \quad (5)$$

Аналогично были рассмотрены все восемь вариантов привязки и размещения. Далее проводится сравнение простых импликант для каждого варианта размещения, и определяются минимальные по Квайну импликанты. В свою очередь выбранные оптимальные по размещению варианты разделим на подварианты, отличающиеся кодированием минтермов.

При кодировании минтермов следует исходить как из условий толерантности в подмножествах минтермов, так и из условия реализации выходной функции с минимальным числом дополнительных инверторов при разработке и создании схемы ЦА ПП на типовых микросхемах счётчиков.

Для установления характера кодирования вариантов настройки схемы, выполненной на суммирующем счётчике, запишем минимальную ДНФ функции выхода для выбранного варианта привязки и размещения, выполнив операцию логического сложения простых импликант этого варианта

$$F = Q_3Q_2Q_1Q_0 \vee \tilde{a}_iQ_3Q_2Q_1 \vee \tilde{a}_k\overline{Q}_3\overline{Q}_2\overline{Q}_1 \vee \tilde{a}_j\overline{Q}_3\overline{Q}_2 \vee \tilde{a}_i \times \tilde{a}_k\overline{Q}_3\overline{Q}_1. \quad (6)$$

Нетрудно заметить, что выражение (6) можно привести к безинверсной форме путем выбора определенным образом литералов настроечных переменных: $\tilde{a}_i = a_i$, $\tilde{a}_k = \overline{a}_k$, $\tilde{a}_j = \overline{a}_j$. Тогда функцию выхода для схемы, реализованной на суммирующем счетчике, можно описать как:

$$F = P_4 \vee a_iQ_3Q_2Q_1 \vee \overline{a}_k\overline{Q}_3\overline{Q}_2\overline{Q}_1 \vee \overline{a}_j\overline{Q}_3\overline{Q}_2 \vee \overline{a}_i \times \overline{a}_k\overline{Q}_3\overline{Q}_1,$$

а преобразуя её по правилу де Моргана, получим безинверсную форму:

$$F = P_4 \vee a_i Q_3 Q_2 Q_1 \vee \overline{a_k \vee Q_3 \vee Q_2 \vee Q_1} \vee \overline{a_j \vee Q_3 \vee Q_2} \vee \overline{a_i \vee a_k \vee Q_3 \vee Q_1}, \qquad (7)$$

где P_4 – представляет собой функцию переноса суммирующего счетчика.

Так как аргументы a_i, a_k, a_j являются элементами множества настроечных переменных $\{a_2, a_1, a_0\}$, то количество вариантов оптимального кодирования (табл. 1), приводящих к нахождению функции выхода в безинверсной форме, будет равно числу перестановок из трех.

Таблица 1
Table 1

Варианты оптимального кодирования
Optimal coding options

	1	2	3	4	5	6
$\tilde{a}_i$	a_0	a_0	a_1	a_1	a_2	a_2
$\tilde{a}_k$	$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_0$	$\bar{a}_0$	$\bar{a}_0$	$\bar{a}_1$
$\tilde{a}_j$	$\bar{a}_2$	$\bar{a}_1$	$\bar{a}_2$	$\bar{a}_2$	$\bar{a}_1$	$\bar{a}_0$

Алгоритм нахождения кодов настройки на заданные режимы формируется на основании индексов минтермов (от p до δ), равных десятичному эквиваленту этого кода. Были рассмотрены три версии алгоритма. С целью сокращения объема проиллюстрируем методику на одной из версий алгоритма, развивая ее до версии, пригодной для программной реализации.

В основу этой версии алгоритма положено свойство толерантности подмножеств P_0, P_1, P_2, сформулированное выше, и общее свойство минтермов, согласно которому произведение разнотипных минтермов, образуемых литералами одних и тех же переменных, тождественно равно нулю.

Приведенные свойства позволяют определить индексы любого из минтермов (от m_p до m_δ), а значит, и коды настройки, рассматривая каждый минтерм, образуемый литералами настроечных переменных (от $m_0 = \bar{a}_2\bar{a}_1\bar{a}_0$ до $m_7 = a_2a_1a_0$), в виде произведения логических сумм выделенных групп (или отрицаний). При умножении не равными нулю будут только произведения, где сомножители являются однотипными минтермами – $m_i \times m_i \times m_i$, следовательно, для нахождения такого произведения достаточно рассмотреть логические суммы в каждой из трех скобок и установить наличие в них минтермов с одинаковыми индексами, что и определяет значение произведения.

Осуществим кодирование минтермов для схемы, выполненной на суммирующем счетчике. Алгоритм идентичен для любого из шести вариантов кодирования, приведенных в таблице 1, поэтому проанализируем только один вариант, для которого: $\tilde{a}_i = a_0 = m_q \vee m_s \vee m_\beta \vee m_\delta$; $\tilde{a}_k = \bar{a}_1 = m_r \vee m_s \vee m_\gamma \vee m_\delta$; $\tilde{a}_j = \bar{a}_2 = m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta$. Поскольку для представления всех минтермов, образуемых литералами трех настроечных переменных, требуются как прямые, так и их инверсные значения, то прежде всего необходимо записать недостающие три $(\bar{a}_0, a_1, a_2)$, выполняя операцию инверсии над соответствующими литералами. При выполнении этой операции используем одно из свойств множества минтермов от одних и тех же переменных, заключающееся в том, что инверсия логической суммы, образованной некоторыми минтермами, равна логической сумме минтермов, дополняющих инвертируемые до единицы. Отсюда следует, что недостающие в первом варианте кодирования литералы можно записать следующим образом:

$$\begin{cases} \overline{a}_0 = \overline{m_q \vee m_s \vee m_\beta \vee m_\delta} = m_p \vee m_r \vee m_\alpha \vee m_\gamma; \\ a_1 = (\overline{\overline{a}_1}) = \overline{m_r \vee m_s \vee m_\gamma \vee m_\delta} = m_p \vee m_q \vee m_\alpha \vee m_\delta;. \\ a_2 = (\overline{\overline{a}_2}) = \overline{m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta} = m_p \vee m_q \vee m_r \vee m_s. \end{cases} \quad (8)$$

Представляем каждый из восьми минтермов, образованных литералами настроечных переменных, в виде произведения соответствующих логических сумм, выделяя в каждой из трех скобок общий минтерм, который и будет равняться представленному:

$$\begin{cases} m_0 = \overline{a}_2\overline{a}_1\overline{a}_0 = (m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta)(m_r \vee m_s \vee m_\gamma \vee m_\delta)(m_p \vee m_r \vee m_\alpha \vee m_\gamma) = m_\gamma; \\ m_1 = \overline{a}_2\overline{a}_1 a_0 = (m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta)(m_r \vee m_s \vee m_\gamma \vee m_\delta)(m_q \vee m_s \vee m_\beta \vee m_\delta) = m_\delta; \\ m_2 = \overline{a}_2 a_1\overline{a}_0 = (m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta)(m_p \vee m_q \vee m_\alpha \vee m_\beta)(m_p \vee m_r \vee m_\alpha \vee m_\gamma) = m_\alpha; \\ m_3 = \overline{a}_2 a_1 a_0 = (m_\alpha \vee m_\beta \vee m_\gamma \vee m_\delta)(m_p \vee m_q \vee m_\alpha \vee m_\beta)(m_q \vee m_s \vee m_\beta \vee m_\delta) = m_\beta; \\ m_4 = a_2\overline{a}_1\overline{a}_0 = (m_p \vee m_q \vee m_r \vee m_s)(m_r \vee m_s \vee m_\gamma \vee m_\delta)(m_p \vee m_r \vee m_\alpha \vee m_\gamma) = m_r; \\ m_5 = a_2\overline{a}_1 a_0 = (m_p \vee m_q \vee m_r \vee m_s)(m_r \vee m_s \vee m_\gamma \vee m_\delta)(m_q \vee m_s \vee m_\beta \vee m_\delta) = m_s; \\ m_6 = a_2 a_1\overline{a}_0 = (m_p \vee m_q \vee m_r \vee m_s)(m_p \vee m_q \vee m_\alpha \vee m_\beta)(m_p \vee m_r \vee m_\alpha \vee m_\gamma) = m_p; \\ m_7 = a_2 a_1 a_0 = (m_p \vee m_q \vee m_r \vee m_s)(m_p \vee m_q \vee m_\alpha \vee m_\beta)(m_q \vee m_s \vee m_\beta \vee m_\delta) = m_q. \end{cases} \quad (9)$$

Аналогичным образом можно найти кодирование минтермов для остальных пяти вариантов.

Опуская подобные выкладки, выполненные для других вариантов, табулируем варианты кодирования в форме десятичных эквивалентов кодов настройки, образуемых соответствующими наборами настроечных переменных, и составим результирующую таблицу для всех режимов настройки (табл. 2). В таблице 3 приведены коды первого варианта настройки, представленные значениями настроечных переменных a_2, a_1, a_0.

Таблица 2
Table 2

Варианты кодирования в виде десятичных эквивалентов кодов настройки
Decimal equivalent encoding options for tuning codes

L	1	2	3	4	5	6	7	8
N_0	p	q	r	s	α	β	γ	δ
1	6	7	4	5	2	3	0	1
2	6	7	2	3	4	5	0	1
3	5	7	4	6	1	3	0	2
4	5	7	1	3	4	6	0	2
5	3	7	2	6	1	5	0	4
6	3	7	1	5	2	6	0	4

Таблица 3
Table 3

Коды первого варианта настройки
First setting codes

L	1	2	3	4	5	6	7	8
a_2	1	1	1	1	0	0	0	0
a_1	1	1	0	0	1	1	0	0
a_0	0	1	0	1	0	1	0	1

Аналогичным образом можно табулировать значения настроечных переменных для остальных пяти вариантов настройки.

Рассмотренная версия алгоритма нахождения кодов настройки предельно проста, но довольно громоздка, особенно при большом числе настроечных переменных r, поскольку необходимо записывать и проводить анализ 2^r произведений для каждого из $r!$ вариантов кодирования. Используя результаты, полученные в этой версии алгоритма на промежуточных этапах, дополним их действиями, позволяющими в целом получить алгоритм, пригодный для программной реализации нахождения кода настройки.

В этом случае множество всех минтермов, определяющих режим настройки, представляем в виде массива, образованного z' столбцами и z'' строками. Общее число элементов массива, которое можно определить как произведение числа строк на число столбцов равно множеству всех минтермов, т. е. 2^r. Если число настроечных переменных (r) – четное, то массив представляет собой квадрат с числом столбцов, равным числу строк, которое можно определить как $z' = z'' = 2^r/2$. Если r – нечетное, то массив можно организовать двумя способами: в виде прямоугольника с «широким основанием» и прямоугольника с «узким основанием». Для первой формы $z' = 2^{\frac{r+1}{2}}, z'' = 2^{\frac{r-1}{2}}$, для второй $z' = 2^{\frac{r-1}{2}}, z'' = 2^{\frac{r+1}{2}}$. Никаких преимуществ одна форма перед другой не имеет.

Элементы массива нумеруются десятичными числами слева направо и вниз, начиная с нуля и заканчивая $2^r - 1$. Каждому элементу массива соответствует минтерм, определяющий кратность настройки. При этом элементу с нулевым номером соответствует минтерм, определяющий коэффициент кратности $L = 1$, а элементу с номером $2^r - 1$ – коэффициент кратности $L = 2^r$. Отсюда следует соотношение между коэффициентами кратности L и номером элемента $N: L = N + 1$.

Если ввести термин четности и нечетности не только для отдельных строк и столбцов, но и для последовательно идущих групп, состоящих из двух, четырех, восьми и в общем случае из 2^r столбцов (строк), считая слева направо для множества столбцов и сверху вниз для строк, то можно будет определить все r подмножеств, содержащих 2^{r-1} минтермов, удовлетворяющих сформулированным условиям толерантности. Такими будут подмножества образованного соответствующими группами столбцов и строк массива: подмножество минтермов, входящих во все нечетные столбцы; подмножество минтермов, входящих во все нечетные пары столбцов; подмножество минтермов, входящих во все нечетные четверки столбцов, и так далее по столбцам вплоть до подмножества, образованного правой половиной всех столбцов; подмножество минтермов, входящих во все нечетные строки; подмножество минтермов, входящих во все нечетные пары строк; подмножество минтермов, входящих во все нечетные четверки строк, и так далее по строкам, вплоть до подмножества, образованного нижней половиной всех строк массива.

Аналогичные определения можно дать для инверсных подмножеств, но только теперь будут фигурировать четные столбцы и четные строки и их группы.

При программной реализации алгоритма наличие тех или других минтермов в рассматриваемом подмножестве в соответствующих элементах массива отмечается единицей, а отсутствие – нулем. Тогда любое из r прямых подмножеств будет определено массивом, содержащим в половине столбцов или строк единицы, а в половине нули. При этом каждому прямому подмножеству массива будет соответствовать инверсное ему подмножество (массив), отличающееся противоположным значением в его элементах.

Поскольку каждому из подмножеств можно поставить в соответствие тот или другой литерал любой настроечной переменной, то это дает возможность программной реализации алгоритма нахождения кода настройки, представляя минтермы, образуемые литералами настроечных переменных, в виде пересечения соответствующих им массивов.

Анализ предложенной версии алгоритма проведем на примере реализации функции выходов схемы ЦА ПП, выполненного на четырехразрядном суммирующем счетчике, представляя множество минтермов, определяющих режимы настройки, в виде прямоугольного массива индексов минтермов, определяющих коды настройки, содержащего две строки и четыре столбца (рис. 4).

p	q	r	s
α	β	γ	δ

Рис. 4. Массив кодов настройки
Fig. 4. Array of setup codes

Разбиваем множество всех минтермов на три подмножества (P_0, P_1, P_2), удовлетворяющих заданным условиям толерантности, представляя каждое из них в виде массива:

$$P_0: \begin{matrix} - & q & - & s \\ - & \beta & - & \delta \end{matrix};\ P_1: \begin{matrix} - & - & r & s \\ - & - & \gamma & \delta \end{matrix};\ P_2: \begin{matrix} - & - & - & - \\ \alpha & \beta & \gamma & \delta \end{matrix}.$$

Каждому из приведенных подмножеств ставим в соответствие инверсное ему подмножество (дополнение)

$$\overline{P}_0: \begin{matrix} p & - & r & - \\ \alpha & - & \gamma & - \end{matrix};\ \overline{P}_1: \begin{matrix} p & q & - & - \\ \alpha & \beta & - & - \end{matrix};\ \overline{P}_2: \begin{matrix} p & q & r & s \\ - & - & - & - \end{matrix}.$$

При программной реализации алгоритма вхождение того или другого минтерма в каждое из подмножеств отмечается единицей, а отсутствие – нулём. В результате получаем массивы, половина элементов которых содержит единицы, а половина – нули.

$$P_0: \begin{matrix} 0101 \\ 0101 \end{matrix};\ \overline{P}_0: \begin{matrix} 1010 \\ 1010 \end{matrix};\ P_1: \begin{matrix} 0011 \\ 0011 \end{matrix};\ \overline{P}_1: \begin{matrix} 1100 \\ 1100 \end{matrix};\ P_2: \begin{matrix} 0000 \\ 1111 \end{matrix};\ \overline{P}_2: \begin{matrix} 1111 \\ 0000 \end{matrix}.$$

В соответствии с рассмотренным вариантом кодирования (табл.1, первый столбец) подмножеству P_0, соответствует a_0, подмножеству P_1 – $\overline{a}_1$, подмножеству P_2 – $\overline{a}_2$. Представим эти соответствия массивами:

$$a_0: \begin{matrix} 0101 \\ 0101 \end{matrix};\ \overline{a}_0: \begin{matrix} 1010 \\ 1010 \end{matrix};\ \overline{a}_1: \begin{matrix} 0011 \\ 0011 \end{matrix};\ a_1: \begin{matrix} 1100 \\ 1100 \end{matrix};\ \overline{a}_2: \begin{matrix} 0000 \\ 1111 \end{matrix};\ a_2: \begin{matrix} 1111 \\ 0000 \end{matrix}.$$

Каждый минтерм, образуемый литералами настроечных переменных, представляем в виде пересечения соответствующих массивов:

$$\overline{a}_2\overline{a}_1\overline{a}_0 = \begin{matrix} 0000 \\ 1111 \end{matrix} \cap \begin{matrix} 0011 \\ 0011 \end{matrix} \cap \begin{matrix} 1010 \\ 1010 \end{matrix} = \begin{matrix} 0000 \\ 0010 \end{matrix} - \gamma;\ \overline{a}_2\overline{a}_1 a_0 = \begin{matrix} 0000 \\ 1111 \end{matrix} \cap \begin{matrix} 0011 \\ 0011 \end{matrix} \cap \begin{matrix} 0101 \\ 0101 \end{matrix} = \begin{matrix} 0000 \\ 0001 \end{matrix} - \delta;$$

$$\overline{a}_2 a_1\overline{a}_0 = \begin{matrix} 0000 \\ 1111 \end{matrix} \cap \begin{matrix} 1100 \\ 1100 \end{matrix} \cap \begin{matrix} 1010 \\ 1010 \end{matrix} = \begin{matrix} 0000 \\ 1000 \end{matrix} - \alpha;\ \overline{a}_2 a_1 a_0 = \begin{matrix} 0000 \\ 1111 \end{matrix} \cap \begin{matrix} 1100 \\ 1100 \end{matrix} \cap \begin{matrix} 0101 \\ 0101 \end{matrix} = \begin{matrix} 0000 \\ 0100 \end{matrix} - \beta;$$

$$a_2\overline{a}_1\overline{a}_0 = \begin{matrix} 1111 \\ 0000 \end{matrix} \cap \begin{matrix} 0011 \\ 0011 \end{matrix} \cap \begin{matrix} 1010 \\ 1010 \end{matrix} = \begin{matrix} 0010 \\ 0000 \end{matrix} - r;\ a_2\overline{a}_1 a_0 = \begin{matrix} 1111 \\ 0000 \end{matrix} \cap \begin{matrix} 0011 \\ 0011 \end{matrix} \cap \begin{matrix} 0101 \\ 0101 \end{matrix} = \begin{matrix} 0001 \\ 0000 \end{matrix} - s;$$

$$a_2 a_1\overline{a}_0 = \begin{matrix} 1111 \\ 0000 \end{matrix} \cap \begin{matrix} 1100 \\ 1100 \end{matrix} \cap \begin{matrix} 1010 \\ 1010 \end{matrix} = \begin{matrix} 1000 \\ 0000 \end{matrix} - p;\ a_2 a_1 a_0 = \begin{matrix} 1111 \\ 0000 \end{matrix} \cap \begin{matrix} 1100 \\ 1100 \end{matrix} \cap \begin{matrix} 0101 \\ 0101 \end{matrix} = \begin{matrix} 0100 \\ 0000 \end{matrix} - q.$$

Анализируя полученные пересечения, можем сделать вывод: минтерму настроечных переменных m_0 соответствует настройка на режим γ, аналогично для других минтермов получим соответствия: $m_1 - \delta;\ m_2 - \alpha;\ m_3 - \beta;\ m_4 - r;\ m_5 - s;\ m_6 - p;\ m_7 - q$.

Поскольку простые импликанты, получаемые для любого из шести вариантов кодирования, по сложности эквивалентны, то схему ЦА ПП, выполненную на типовом

суммирующем счетчике, приведем только для одного варианта кодирования (рис. 5), представляя функцию выхода в виде:

$$F = P_4 \vee a_0 Q_3 Q_2 Q_1 \vee \overline{a_1 \vee Q_3 \vee Q_2 \vee Q_1} \vee \overline{a_2 \vee Q_3 \vee Q_2} \vee \overline{a_2 \vee a_1 \vee Q_3 \vee Q_1}\,. \qquad (10)$$

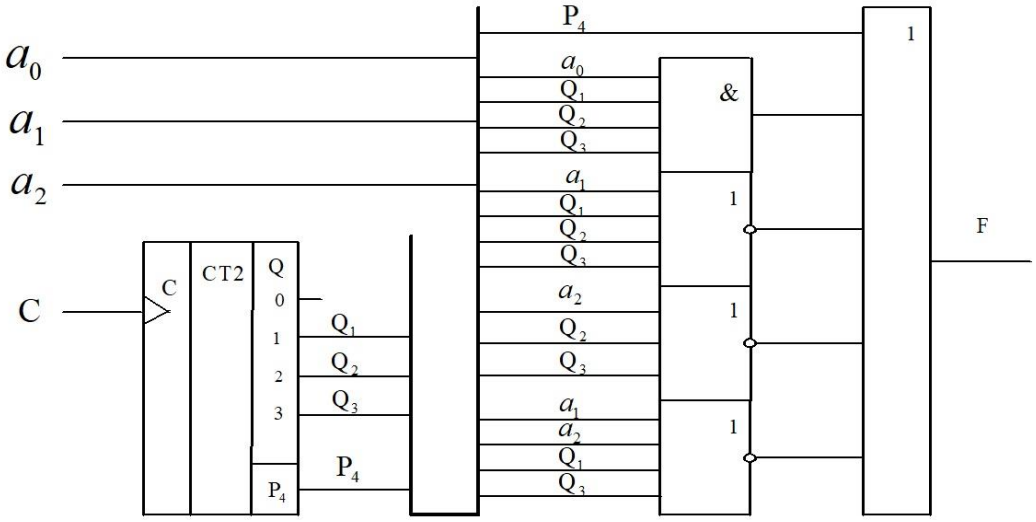

Рис. 5. Схема ЦА ПП, выполненная на типовом суммирующем счетчике
Fig. 5. DAC circuit PP made on a typical totalizing counter

По такой же методике построена схема формирователя импульсной последовательности с перестраиваемыми параметрами, оригинальность которой подтверждена патентом РФ на изобретение [Рубанов и др., 2020], в котором подробно описан принцип действия устройства.

Решение задач оптимального синтеза цифровых автоматов Мили с перестраиваемыми параметрами выходных сигналов связано с перебором большого числа вариантов. Представление функций в форме обобщенных логических функций позволило значительно уменьшить это число и, как следствие, сократить время проектирования. Однако процесс проектирования, как видно, требует хороших навыков работы с обобщенными функциями на различных этапах проекта.

Заключение

Предложенный метод синтеза формирователей тестовой последовательности с перестраиваемыми параметрами целесообразно использовать при проектировании цифровых автоматов с гибко изменяемым тестовым контентом, когда сначала требуется детерминистический тест для распознавания подозрительных узлов, а затем генерируются дополнительные ориентированные тесты, расширяющие тестовый контент с целью уточнения характера неисправности.

Метод синтеза, основанный на использовании обобщенных логических функций, является более универсальным с точки зрения широты его применения для проектирования цифровых автоматов любого назначения, при этом он позволяет не только упростить процедуру нахождения простых импликант, но и обеспечивает контроль достоверности и минимальности полученного результата.

Следует также отметить, что при решении задач оптимального синтеза цифровых автоматов Мили с перестраиваемыми параметрами выходных сигналов, использование представления функций в форме ОЛФ значительно уменьшает число вариантов перебора по сравнению с классическим подходом.

Список литературы

1. Бессонов А.А., Стешкович Н.Т., Турчина Е.Д. 1986. Автоматизация построения контролирующих тестов. Я., Энергия, 224.

2. Бондаренко М.Ф., Кривуля Г.Ф., Рябцев В.Г., Фрадков С.А., Хаханов В.И. 2000. Проектирование и диагностика компьютерных систем и сетей. К., НМЦ ВО, 306.

3. Горовой А.А., Ващевский В.Ф., Доценко Б.И., Рубанов В.Г., Черняк С.П. 1990. Микропроцессорные агрегатные комплексы для диагностирования технических систем. К., Техника, 168.

4. Рубанов В.Г., Коробкова Е.Н. 2008. Логическое проектирование цифровых устройств, основанное на представлении функций в обобщенной форме. Белгород, Изд-во БГТУ, 335.

5. Рубанов В.Г., Коробкова Е.Н. 2009. Методы анализа и синтеза цифровых устройств (проектирование цифровых элементов автоматики и вычислительной техники). Белгород, Изд-во БГТУ, 291.

6. Рубанов В.Г. 2011. Системный подход к проектированию управляемых мобильных логических средств, обладающих свойством живучести. 176–187.

7. Рубанов В.Г., Коробкова Е.Н., Кариков Е.Б. 2020. Формирователь периодической последовательности импульсов. Патент РФ №2019124598. Бюл. 12.

8. Тюрев С.Ф. 2004. Функционально-полные толерантные булевы функции и цифровые схемы на их основе. Пермь, ПГСА, 118.

9. Чжен Г., Меннинг Г. Метц Г. 1972. Диагностика отказов цифровых вычислительных систем. М., Мир, 232 (Zheng G., Menning G., Metz G. 1972. Diagnostics of failures of digital computing systems. IEEE Transactions on Systems Man and Cybernetics 3 (3): 301–301).

10. Amyeen M. Enamul, Kim Dongok, Chandrasekar Maheshwar, Noman Mohammad, Venkataraman Srikanth, Jain Anurag, Goel Neha, Sharma Ramesh. 2016. A novel diagnostic test generation methodology and its application in production failure isolation, in 2016 IEEE International Test Conference (ITC), 15–17.

11. Bartenstein T. 2000. Fault Distinguishing Pattern Generation, in Proc. IEEE Int. Test Conf.

12. Bernardi P., et. al. 2008. An Effective Technique for the Automatic Generation of Diagnosis-Oriented Programs for Processor Cores, in Proc. IEEE Trans. on CAD of Integrated Circuits and Systems.

13. Bhatti N.K, Blanton R.D. 2006. Diagnostic Test Generation for Arbitrary Faults, in Proc. IEEE Int. Test Conf.

14. Chandrasekar M., Hsiao M.S. 2009. Diagnostic Test Generation for silicon diagnosis with an incremental learning framework based on search state compatibility, in Proc. IEEE High Level Design Validation and Test Workshop, 68–75.

15. Chandrasekar M., Rahagude N.P., Hsiao M.S. 2010. Search State Compatibility based Incremental Learning Framework and output deviation based X-filling for diagnostic test generation, Springer Journal of Electronic Testing, vol. 26, no. 2, 165–176.

16. Corno F., Prinetto P., Rebaudengo M., Reorda M. Sonza. 1995. GARDA: A Diagnostic ATPG for Large Synchronous Sequential Circuits, in Proc. European Design and Test Conf., 267–271.

17. Desineni R., Poku O., Blanton R.D. 2006. A logic diagnosis methodology for improved localization and extraction of accurate defect behavior, in Test Conference, ITC'06. IEEE International. IEEE.

18. Girard P., Landrault G., Pravossoudovitch S., Rodriguez B. 1996. A Diagnostic ATPG for Delay Faults based on Genetic Algorithm, in Proc. IEEE Int. Test Conf.

19. Gruning T., Mahlstedt U., Koopmeiners H. 1991. DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits, in Proc. IEEE Int. Conf. Computer-Aided Design, 194–197.

20. Li Z., Goel S.K., Lee F., Chakrabarty K. 2015. Efficient observationpoint insertion for diagnosability enhancement in digital circuits, in Proc. IEEE Int. Test Conf., 1–10.

21. Lin Y.-T., Poku O., Bhatti N.K., Blanton R.D. 2008. Physically-aware n-detect test pattern selection, in Proceedings of the conference on Design, automation and test in Europe. ACM, 634–639.

22. Medina P. Camurati D., Prinetto P., Reorda M. Sonza. 1990. A Diagnostic Test Pattern Generation Algorithm, in Proc. IEEE Int. Test Conf., 52–58.

23. Pomeranz I. 2012. Gradual diagnostic test generation and observation point insertion based on the structural distance between indistinguished fault pairs IEEE Trans. Very Large Scale Integr.(VLSI) Syst., vol. 20, no. 6, 1026–1035.

24. Riefert A., Sauer M., Reddy S., Becker B. 2015. Improving diagnosis resolution of a fault detection test set, in VLSI Test Symposium (VTS) IEEE 33rd. IEEE.

25. Tanwir Sarmad, Hsiao Michael S., Lingappan Loganathan. 2016. Hardware-in-the-loop Model-Less Diagnostic Test Generation, in 2016 IEEE International High Level Design Validation and Test Workshop (HLDVT), 128–133.

26. Wang S., Wei W. 2009. Machine learning-based volume diagnosis, in Design, Automation & Test in Europe Conference & Exhibition, 902–905.

27. Wu Cheng-Hung, Lin Sheng-Lin, Lee Kuen-Jong, Reddy Sudhakar M. 2018. A Repair-for-Diagnosis Methodology for Logic Circuits, in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2254–2267.

28. Xue Y., Poku O., Li X., Blanton R.D. 2013. Padre: physically-aware diagnostic resolution enhancement, in Test Conference (ITC), IEEE International, IEEE.

29. Yu X., Blanton R. 2012. Diagnosis-assisted adaptive test, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactionson, vol. 31, no. 9, 1405–1416.

References

1. Bessonov A.A., Steshkovich N.T., Turchina E.D. 1986. Avtomatizacija postroenija kontrolirujushhih testov [Automation of building control tests]. Ja., Jenergija, 224. (in Russia)

2. Bondarenko M.F., Krivulja G.F., Rjabcev V.G., Fradkov S.A., Hahanov V.I. 2000. Proektirovanie i diagnostika komp'juternyh sistem i setej [Design and diagnostics of computer systems and networks]. K., NMC VO, 306. (in Russia)

3. Gorovoj A.A., Vashhevskij V.F., Docenko B.I., Rubanov V.G., Chernjak S.P. 1990. Mikroprocessornye agregatnye kompleksy dlja diagnostirovanija tehnicheskih sistem [Microprocessor-based aggregate complexes for diagnosing technical systems]. K., Tehnika, 168. (in Russia)

4. Rubanov V.G., Korobkova E.N. 2008. Logicheskoe proektirovanie cifrovyh ustrojstv, osnovannoe na predstavlenii funkcij v obobshhennoj forme [Logical design of digital devices based on the representation of functions in a generalized form]. Belgorod, Izd-vo BGTU, 335. (in Russia)

5. Rubanov V.G., Korobkova E.N. 2009. Metody analiza i sinteza cifrovyh ustrojstv (proektirovanie cifrovyh jelementov avtomatiki i vychislitel'noj tehniki): ucheb. posobie [Methods of analysis and synthesis of digital devices (design of digital elements of automation and computer technology)]. Belgorod, Izd-vo BGTU, 291. (in Russia)

6. Rubanov V.G. 2011. Sistemnyj podhod k proektirovaniju upravljaemyh mobil'nyh logicheskih sredstv , obladajushhih svojstvom zhivuchesti [Systematic approach to the design of managed mobile logical means having the property of vitality].176–187 (in Russia)

7. Rubanov V.G., Korobkova E.N., Karikov E.B. 2020. Periodic flash generator. Patent RF №2019124598. Bull. 12. (in Russia)

8. Tjurev S.F. 2004. Funkcional'no-polnye tolerantnye bulevy funkcii i cifrovye shemy na ih osnove [Functionally-complete tolerance Boolean functions and digital circuits based on them]. Perm', PGSA, 118. (in Russia)

9. Chzhen G., Menning G. Metc G. 1972. Diagnostika otkazov cifrovyh vychislitel'nyh sistem. M., Mir, 232. (Zheng G., Menning G., Metz G. 1972. Diagnostics of failures of digital computing systems. IEEE Transactions on Systems Man and Cybernetics 3(3):301–301).

10. Amyeen M. Enamul, Kim Dongok, Chandrasekar Maheshwar, Noman Mohammad, Venkataraman Srikanth, Jain Anurag, Goel Neha, Sharma Ramesh. 2016. A novel diagnostic test generation methodology and its application in production failure isolation, in 2016 IEEE International Test Conference (ITC), 15–17.

11. Bartenstein T. 2000. Fault Distinguishing Pattern Generation, in Proc. IEEE Int. Test Conf.

12. Bernardi P., et. al. 2008. An Effective Technique for the Automatic Generation of Diagnosis-Oriented Programs for Processor Cores, in Proc. IEEE Trans. on CAD of Integrated Circuits and Systems.

13. Bhatti N.K, Blanton R.D. 2006. Diagnostic Test Generation for Arbitrary Faults, in Proc. IEEE Int. Test Conf.

14. Chandrasekar M., Hsiao M.S. 2009. Diagnostic Test Generation for silicon diagnosis with an incremental learning framework based on search state compatibility, in Proc. IEEE High Level Design Validation and Test Workshop, 68–75.

15. Chandrasekar M., Rahagude N.P., Hsiao M.S. 2010. Search State Compatibility based Incremental Learning Framework and output deviation based X-filling for diagnostic test generation, Springer Journal of Electronic Testing, vol. 26, no. 2, 165–176.

16. Corno F., Prinetto P., Rebaudengo M., Reorda M. Sonza. 1995. GARDA: A Diagnostic ATPG for Large Synchronous Sequential Circuits, in Proc. European Design and Test Conf., 267–271.

17. Desineni R., Poku O., Blanton R.D. 2006. A logic diagnosis methodology for improved localization and extraction of accurate defect behavior, in Test Conference, ITC'06. IEEE International. IEEE.

18. Girard P., Landrault G., Pravossoudovitch S., Rodriguez B. 1996. A Diagnostic ATPG for Delay Faults based on Genetic Algorithm, in Proc. IEEE Int. Test Conf.

19. Gruning T., Mahlstedt U., Koopmeiners H. 1991. DIATEST: A Fast Diagnostic Test Pattern Generator for Combinational Circuits, in Proc. IEEE Int. Conf. Computer-Aided Design, 194–197.

20. Li Z., Goel S.K., Lee F., Chakrabarty K. 2015. Efficient observationpoint insertion for diagnosability enhancement in digital circuits, in Proc. IEEE Int. Test Conf., 1–10.

21. Lin Y.-T., Poku O., Bhatti N.K., Blanton R.D. 2008. Physically-aware n-detect test pattern selection, in Proceedings of the conference on Design, automation and test in Europe. ACM, 634–639.

22. Medina P. Camurati D., Prinetto P., Reorda M. Sonza. 1990. A Diagnostic Test Pattern Generation Algorithm, in Proc. IEEE Int. Test Conf., 52–58.

23. Pomeranz I. 2012. Gradual diagnostic test generation and observation point insertion based on the structural distance between indistinguished fault pairs IEEE Trans. Very Large Scale Integr.(VLSI) Syst., 20 (6): 1026–1035.

24. Riefert A., Sauer M., Reddy S., Becker B. 2015. Improving diagnosis resolution of a fault detection test set, in VLSI Test Symposium (VTS) IEEE 33rd. IEEE.

25. Tanwir Sarmad, Hsiao Michael S., Lingappan Loganathan. 2016. Hardware-in-the-loop Model-Less Diagnostic Test Generation, in 2016 IEEE International High Level Design Validation and Test Workshop (HLDVT), 128–133.

26. Wang S., Wei W. 2009. Machine learning-based volume diagnosis, in Design, Automation & Test in Europe Conference & Exhibition, 902–905.

27. Wu Cheng-Hung, Lin Sheng-Lin, Lee Kuen-Jong, Reddy Sudhakar M. 2018. A Repair-for-Diagnosis Methodology for Logic Circuits, in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2254–2267.

28. Xue Y., Poku O., Li X., Blanton R.D. 2013. Padre: physically-aware diagnostic resolution enhancement, in Test Conference (ITC), IEEE International, IEEE.

29. Yu X., Blanton R. 2012. Diagnosis-assisted adaptive test, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactionson, vol. 31, no. 9, 1405–1416.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Коробкова Елена Николаевна, кандидат технических наук, доцент, доцент кафедры технической кибернетики БГТУ им. В.Г. Шухова, Белгород, Россия

Луценко Оксана Витальевна, кандидат технических наук, доцент, доцент кафедры стандартизации и управления качеством БГТУ им. В.Г. Шухова, Белгород, Россия

Рубанов Василий Григорьевич, доктор технических наук, профессор, заведующий кафедрой технической кибернетики БГТУ им. В.Г. Шухова, Белгород, Россия

INFORMATION ABOUT THE AUTHORS

Elena N. Korobkova, Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of «Technical Cybernetics» Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia

Oksana V. Lutsenko, Candidate of Technical Sciences, Associate Professor, Associate Professor of the Department of «Standardization and Quality Management» Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia

Vasily G. Rubanov, Doctor of Technical Sciences, Professor, Head of the Department of "Technical Cybernetics" Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia