УДК 004.054

Тестирование ПЛИС с помощью конвейеризированных генераторов контрольных кодов

© О.М. Брехов, М.О. Ратников

Московский авиационный институт (национальный исследовательский университет), Москва, 125993, Россия

Предложен подход к решению задач исследования характеристик ПЛИС и тестирования систем, использующих ПЛИС, на ранних этапах разработки посредством универсальных тестовых прошивок на основе конвейеризированных генераторов контрольных кодов. Разработаны тестовые прошивки: на основе CRC (для выявления одиночных и множественных сбоев или отказов) и на основе кода Хэмминга (для выявления места сбоя или отказа).

Ключевые слова: *ПЛИС, тестирование, отбраковка, окружение ПЛИС, конвейеризированные функции, самокорректирующиеся коды, CRC, код Хэмминга.*

Введение. В процессе разработки систем, основанных на программируемых логических интегральных схемах (ПЛИС), при тестировании самой ПЛИС и ее окружения на некоторых этапах требуется не только выявить факт сбоя (или отказа), но и определить его место. Существующие подходы не гарантируют выявления множественных сбоев, не позволяют точно обнаруживать место сбоя и не отвечают требованиям масштабируемости. Кроме того, они требуют создания отдельной прошивки для каждого этапа разработки.

В данной статье предлагается новый подход к созданию тестовой прошивки, основанный на реализации конвейеризированного генератора контрольных кодов, который позволяет выявлять множественные сбои (или отказы), а также определять место их возникновения.

1. Требования к тестовой прошивке. Определим понятия тестовой и целевой прошивки, используемые в статье. *Прошивка* — двоичный файл, определяющий конфигурацию целевой ПЛИС и получаемый в результате синтеза и трассировки функционального описания системы. *Тестовая прошивка* — прошивка, полученная из тестового функционального описания и предназначенная для выполнения только определенного теста или группы тестов. *Целевая прошивка* — прошивка, полученная из целевого функционального описания и предназначенная для обеспечения выполнения всех требований, перечисленных в техническом задании на разрабатываемую систему.

Введем также понятие *входного вектора* — множество двоичных значений, установленных на входы тестовой системы в данный момент.

Тестовая прошивка необходима на следующих этапах тестирования:

• выбор аппаратной платформы;

• отбраковка ПЛИС;

• тестирование окружения ПЛИС (системной платы и ее подсистем [1, 2].

Существующие подходы, описанные в [3] и [4], обладают следующими недостатками:

• не гарантируют выявления одиночных и многократных сбоев;

• не обеспечивают точного выявления места возникновения сбоев;

• требуют создания отдельной тестовой прошивки для каждого этапа тестирования;

• не всегда обеспечивают корректную обработку множественных сбоев.

К тестовому функциональному описанию предъявляются следующие требования:

1) *синтезируемость* — корректное выполнение этапов синтеза, трассировки и генерации конфигурационного файла для целевой ПЛИС, т. е. не должны нарушаться ограничения, заданные конкретной микросхемой и используемым программным обеспечением;

2) *чувствительность к сбоям и отказам* — обеспечение максимальной чувствительности к сбоям и отказам, включая детекцию многократных сбоев и отказов;

3) *масштабируемость* — быстрое изменение степени логической загрузки ПЛИС;

4) *сохранение логической емкости* — соответствие между предполагаемым значением занятых логических ресурсов и значением, полученным с помощью оценки синтезатора и трассировщика. Это можно обеспечить такими способами, как:

• создание функционального описания в формате, соответствующем этапу постсинтеза, т. е. в виде списка библиотечных элементов и связей между ними. Недостатками этого подхода являются трудоемкость и привязка к конкретной библиотеке элементов;

• отключение оптимизации. Однако для некоторых типов исследования микросхем это недопустимо, так как может негативно сказаться на точности проводимого тестирования;

5) *воспроизводимость* — значения, получаемые в процессе работы тестового устройства, могут быть вычислены заранее и, соответственно, сверены с данными, полученными в результате работы исследуемой ПЛИС.

2. Конвейеризированные генераторы контрольных кодов. Наиболее полно приведенным ранее требованиям удовлетворяют конвейерные системы, в которых каждая стадия представляет собой множество логических элементов, реализующих выбранную синтезируемую функцию, и регистр для хранения вычисленного значения. Такие системы хорошо масштабируются путем увеличения количества стадий конвейера, позволяют не допускать усложнения логических функций, реализованных на каждой из ступеней. В качестве начального значения в такой системе могут быть задействованы значения на входах ПЛИС. Преимуществом конвейерных систем с повторяющимися функциями является их регулярная структура, что значительно упрощает работу по размещению логической нагрузки внутри ПЛИС.

В качестве основной тестовой функции предлагается использовать генераторы контрольных кодов. В роли информационной части кода используется входной битовый массив; на выходах тестовой системы устанавливаются контрольные биты, вычисленные для этого массива. Результатом работы каждой ступени конвейера является промежуточное значение контрольных битов, вычисленное для i битов входного потока, где i — номер данной ступени. Также между стадиями могут передаваться вспомогательные значения-признаки, которые используются для выполнения следующего шага алгоритма.

Входной битовый массив будем называть *входным потоком*. Будем разделять понятия эталонного и фактического входных потоков. Под *эталонным входным потоком* понимается множество двоичных значений, которые были переданы на входы тестируемой системы или должны были быть получены в результате корректной работы внутренних узлов тестируемой подсистемы. *Фактический входной поток* — множество двоичных значений, принятых устройством или полученных от внутренних узлов. Конвейеризированный генератор осуществляет вычисление контрольного кода для фактического входного потока. Появление сбоев в работе устройства ведет к появлению отличий между эталонным и фактическим входными потоками или искажений в контрольном коде. Совокупность эталонного входного потока и соответствующих контрольных битов будем называть *эталонным контрольным кодом*.

Эталонный входной поток может быть представлен константными или динамически изменяющимися значениями. Константные значения устанавливаются на входы тестовой системы перед началом тестирования и не изменяются до его окончания.

Входной поток на основе динамически изменяющихся значений представляет собой массив из множества входных векторов, при этом

на каждом такте на входы тестовой системы устанавливается новый входной вектор. Результатом работы такой тестовой системы является множество наборов контрольных кодов, вычисленных для разных входных векторов. При вычислении контрольных битов для эталонного входного потока будем учитывать особенность работы конвейерных систем с параллельной подачей на каждом такте входных данных на все ступени конвейера.

Пусть у нас имеется эталонный входной поток S. Тогда по определению

$$S = \{V_1, V_2, \ ..., \ V_t\}. \tag{1}$$

Здесь $V_1, \ ..., \ V_t$ — входные векторы, причем

$$V_i = \{b_{i1}, b_{i2}, \ ..., \ b_{in}\}, \tag{2}$$

где $b_{i1}, \ ..., \ b_{in}$ — n бит входного потока, т.е. набор двоичных значений, установленных на входы тестовой системы на i-м такте ее работы.

Пусть R — множество выходных значений тестовой системы, т. е.

$$R = \{R_1, R_2, \ ..., \ R_t\}. \tag{3}$$

Здесь $R_i, \ i = \overline{1,t}, \ $ — наборы контрольных битов в соответствующие моменты времени, причем

$$R_i = \{r_{i1}, r_{i2}, \ ..., \ r_{ik}\}, \tag{4}$$

где $r_{i1}, \ ..., \ r_{ik}$ — контрольные биты, составляющие i-й набор.

По определению

$$R = F(S). \tag{5}$$

Так как в тестовой системе не используются запоминающие устройства (ЗУ), долговременно хранящие данные (в каждый момент времени в ЗУ тестового конвейера могут находиться только промежуточные результаты, вычисленные на предыдущем такте) и на систему наложены определенные ограничения, то R_i зависит от всех входных векторов V, которые устанавливались на входы конвейера за n тактов до i-го такта, где n — длина конвейера. Учитывая, что последний входной вектор, оказавший влияние на R_i, был установлен за один такт до получения R_i, запишем

$$R_i = F(V_{i-n-1}, V_{i-n-1+1}, \ ..., \ V_{i-1}). \tag{6}$$

В силу того что на каждом такте в расчете промежуточного значения R_i участвовал только один бит из входного потока (в случае побитовой обработки входных данных), а каждое следующее промежуточное значение R_i вычислялось на соответствующей ступени (номер ступени, вычисляющей промежуточное значение R_i, на каждом такте увеличивался на единицу), имеем

$$R_i = F\left(b_{(i-n-1)1}, b_{(i-n-1+1)2}, \ \ldots, \ b_{(i-1)n}\right). \qquad (7)$$

Биты входного массива *S*, входящие в состав векторов *V* и участвующие в расчете R_i, приведены на рис. 1.

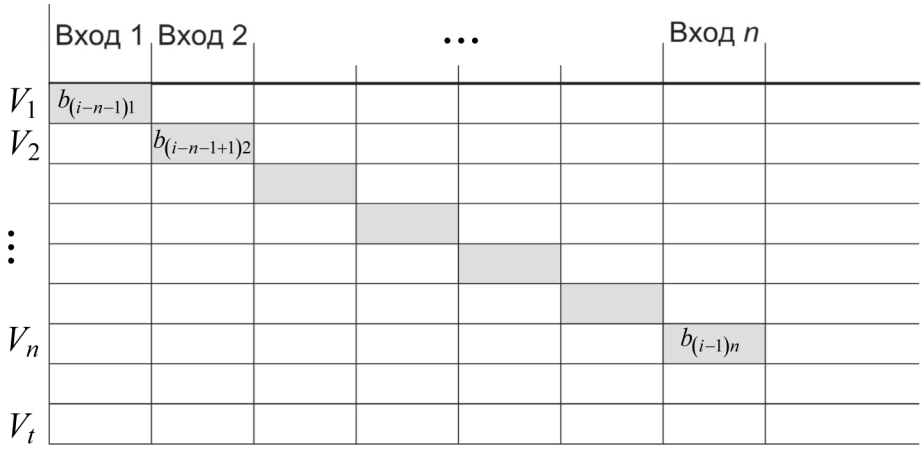

Рис. 1. Входные данные в конвейере с динамическим входным потоком

В задачах тестирования ПЛИС может быть использован любой из алгоритмов генерации контрольных кодов, работающий с поступающим потоком данных. Исходный циклический алгоритм, выполняющий побитовую обработку входного потока, преобразуется в конвейерную структуру, в которой на каждую стадию конвейера подается соответствующий бит входного потока. Основным элементом тестовой системы является блок, реализующий выбранный алгоритм вычисления контрольных кодов.

Для обнаружения места возникновения сбоя предлагается создание прошивки на основе конвейеризированного алгоритма вычисления самокорректирующихся кодов.

Алгоритм обнаружения сбоев:

1. Определить способ генерации и значения битов входного потока.

2. Аналитически или с помощью системы моделирования вычислить контрольные биты для эталонного входного потока.

3. Провести тестирование системы с помощью полученного ранее эталонного потока и получить с выходов тестируемой системы контрольные биты для фактического входного потока.

4. Сравнить контрольные биты, вычисленные для эталонного и фактического входных потоков. В случае расхождения:

4.1. Для генераторов контрольной суммы зафиксировать факт сбоя.

4.2. Для генераторов самокорректирующихся кодов:

• подставить контрольные биты, полученные от тестовой системы, на место контрольных битов в эталонном коде;

• вычислить синдром ошибки для полученного кода;

• на основании вычисленного синдрома ошибки определить номер ступени конвейера, на которой произошел сбой.

5. Сделать выводы о результатах тестирования.

Общий вид такой тестовой системы приведен на рис. 2.

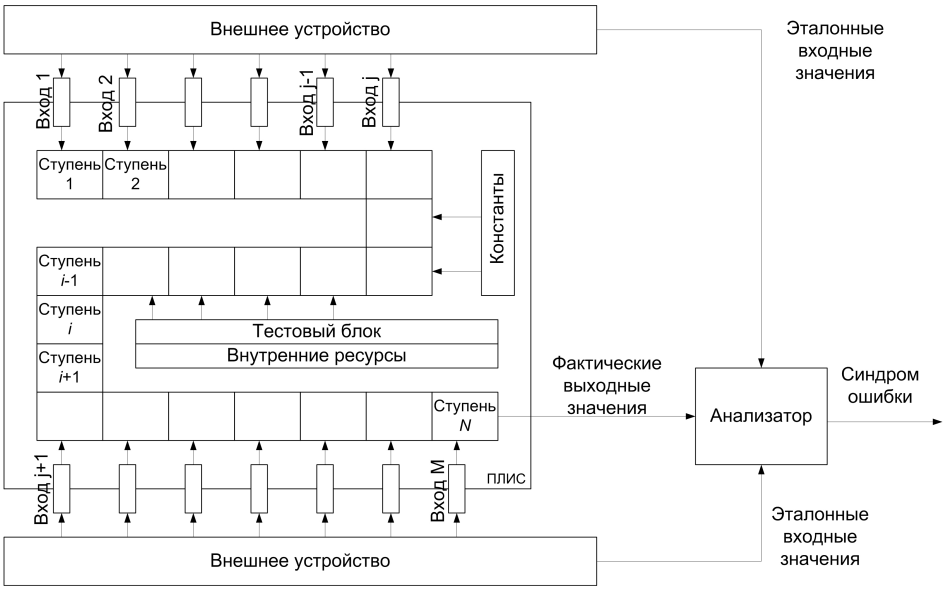

Рис. 2. Общий вид тестовой системы

На рис. 3 приведен общий вид базового и конвейеризированного алгоритмов. Структурные схемы устройств, реализующих базовый и конвейеризированный алгоритмы, приведены соответственно на рис. 4 и 5.

На рис. 6 приведена диаграмма работы конвейеризированной тестовой системы с константными значениями на входах. На диаграмме показаны моменты начала работы (разгон конвейера) и процесс обработки сбоя. Значения R1…R5 — промежуточные результаты работы стадий 1…5 при отсутствии сбоев. Значения F1…F5 — результат работы стадий 1…5 при обработке значения со сбоем.

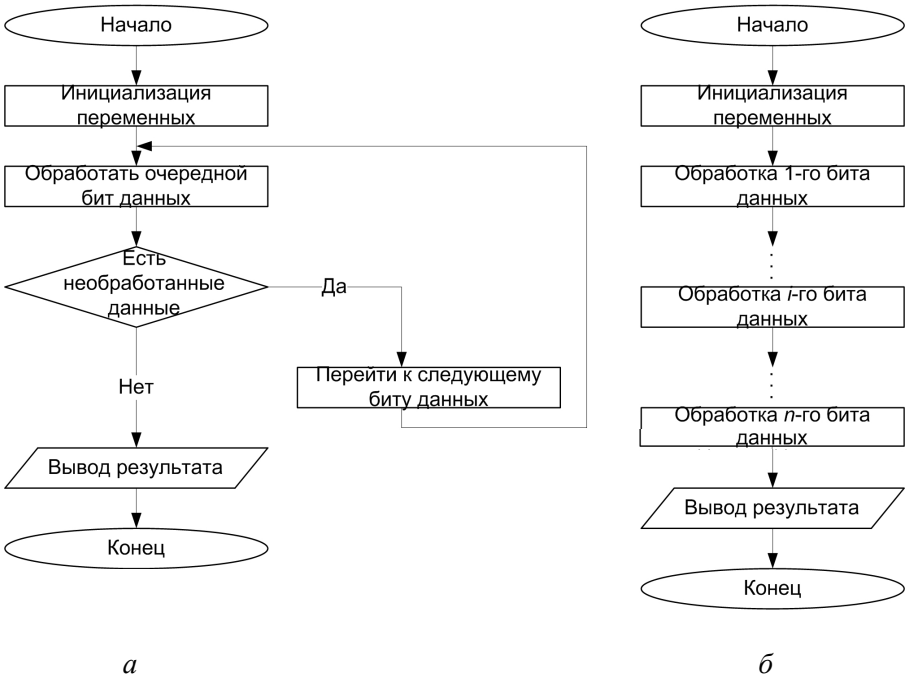

Рис. 3. Базовый (*а*) и конвейеризированный (*б*) алгоритмы

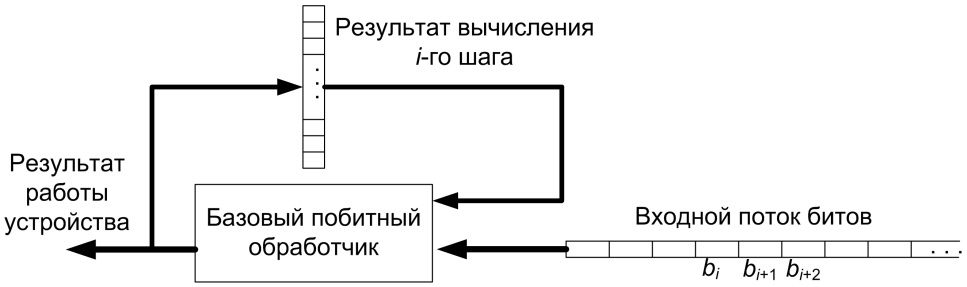

Рис. 4. Схема устройства, реализующего базовый алгоритм

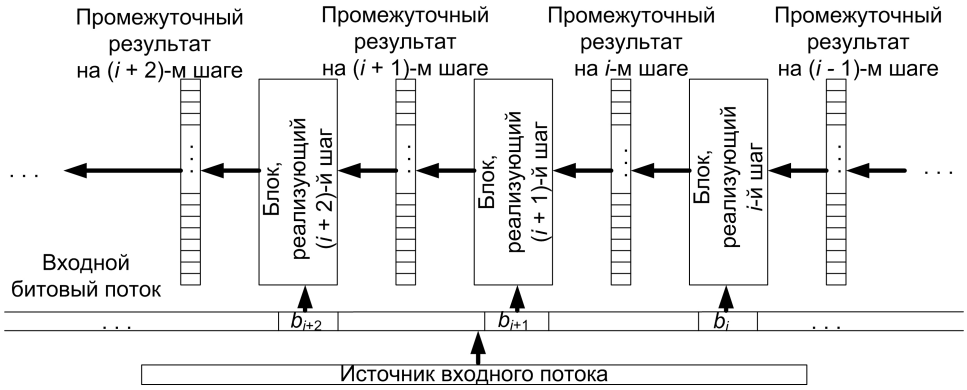

Рис. 5. Схема устройства, реализующего конвейеризированный алгоритм

На выходы тестовой системы подается последнее вычисленное значение (здесь R5). При отсутствии сбоев на каждом такте работы конвейера на выходах тестовой системы будет устанавливаться значение R5, которое сравнивается с эталонным. Признаком отсутствия сбоев в тестовой системе является равенство значения, полученного от тестовой системы, и эталонного. В случае выявления сбоя на выходах тестовой системы будет установлено значение F5, отличное от R5 (не совпадающее с эталонным). После окончания внешнего воздействия работа конвейера будет восстановлена и на выходы тестовой системы снова будет подано значение R5.

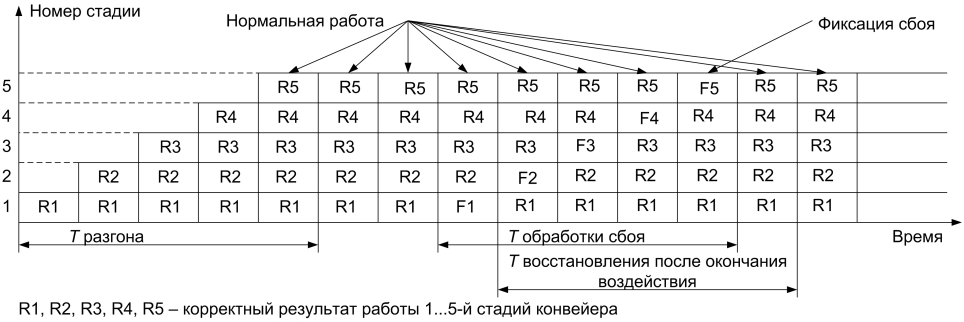

Рис. 6. Диаграмма работы конвейеризированной тестовой системы

2.1. ***Тестовая система на основе конвейеризированного генератора CRC-кода.*** Одним из самых распространенных способов выявления сбоев при передаче данных является алгоритм циклического вычисления контрольного кода CRC. Рассмотрим пример создания тестовой системы на основе алгоритма CRC [5].

На рис. 7 приведена общая схема генератора CRC для битового входного потока. Эта схема обеспечивает обработку входного потока данных, поступающих из блока «Вход». Запоминающие элементы R образуют регистр, в котором хранится текущее значение контрольного кода. При поступлении очередного бита данных из входного потока значение в регистре сдвигается влево на один разряд. Старший бит регистра определяет действие, которое будет произведено на этом шаге: «сдвиг влево» или «сложение по модулю 2 с порождающим полиномом и сдвиг влево». Полученный из входного потока бит после преобразования записывается в младший бит регистра.

Для решения задачи тестирования ПЛИС необходимо обеспечить параллельное получение битов входного потока и их обработку для всех ступеней конвейера. После окончания разгона конвейера на каждом такте тестовая система должна выдавать контрольный код. Для этого каждая ступень конвейера представляется в виде регистра

разрядностью N (длина порождающего полинома), логического элемента «сложение по модулю 2», мультиплексора; порождающий полином задается константой. Входной поток выполняет функцию многочлена $P(x)$.

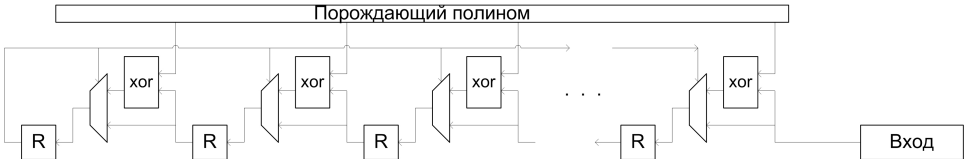

Рис. 7. Схема генерации контрольного кода CRC

Количество битов входного потока совпадает с количеством ступеней конвейера. Во время работы тестовой системы на очередном такте каждая ступень конвейера получает промежуточное значение контрольного кода, определяет и выполняет текущее действие над этим промежуточным значением и соответствующим битом входного потока. После чего очередное промежуточное значение передается на следующую стадию. Структурная схема такого устройства приведена на рис. 8.

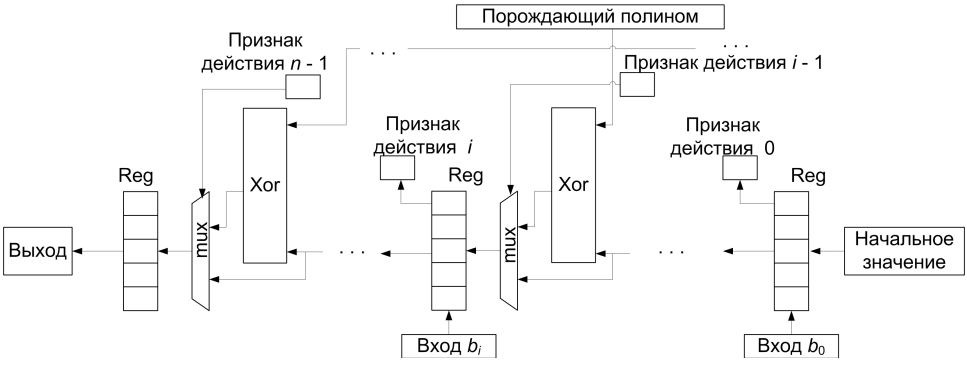

Рис. 8. Развернутая схема генерации CRC

Так как представленная схема является доработанной реализацией схемы генерации CRC, она сохраняет все свойства базовой схемы генерации контрольного кода. В результате работы схемы на последней стадии конвейера, после окончания разгона конвейера устанавливается корректное значение CRC.

Во время процесса тестирования появление любого сбоя в тестовом конвейере в соответствии со свойствами CRC приведет к значительному изменению выходного результата. В зависимости от дальнейшей динамики изменения значения и его продолжительности

можно будет сделать выводы о типе нарушения работы системы (сбой или отказ).

Реализация тестового функционального описания развернутого генератора CRC. В качестве примера реализации тестового функционального описания рассмотрим реализацию конвейера с развернутым алгоритмом генерации контрольного кода CRC. Для исходных кодов тестовой системы был использован язык описания и верификации аппаратного обеспечения SystemVerilog 2005 (IEEE 1800-2005) [6]. Результаты синтеза с помощью пакета Mentor Graphics Precision Synthesis представлены в табл. 1.

Таблица 1

Результаты синтеза развернутого генератора CRC

Количество ступеней	В базис ПЛИС Altera Stratix					В базис ПЛИС Xilinx Virtex		
	Занято ячеек	Частота, МГц	nets	LCs	Inst.	LUTs	CLB	Частота, МГц
100	765	925,926	794	765	784	271	382	698,324
200	1565	819,672	1594	1565	1584	571	782	677,507
300	2365	819,672	2394	2365	2384	871	1182	663,570
400	3165	819,672	3194	3165	3184	1171	1582	663,570
500	3965	819,672	3994	3965	3984	1471	1982	663,570
600	4765	819,672	4794	4765	4784	1771	2382	663,570
700	5565	819,672	5594	5565	5584	2071	2782	663,570
800	6365	819,672	6394	6365	6384	2371	3182	663,570
900	7165	819,672	7194	7165	7184	2671	3582	663,570

Для двух ПЛИС от разных производителей система продемонстрировала идентичные результаты: монотонное увеличение количества занятых ресурсов при увеличении количества ступеней конвейера. Максимальная частота работы системы близка к максимально возможной для соответствующей модели ПЛИС. Это подтверждает выполнение требования масштабируемости.

2.2. ***Тестовая система на основе конвейеризированного генератора кода Хэмминга.*** В качестве примера реализации тестового функционального описания с определением места выявления сбоя рассмотрим реализацию конвейера с развернутым алгоритмом генерации кода Хэмминга, выявляющего две ошибки и исправляющего одну ошибку.

Каждая из стадий конвейера в такой тестовой системе обрабатывает один бит входного потока. В целях сохранения простоты вычисления эталонного значения и упрощения верификации тестовой про-

шивки все ступени конвейера с номерами, равными степени 2 (0, 1, 2, 4, 8, …), не выполняют никаких действий и представляют собой регистр.

Обработка очередного бита входного потока заключается в попарном сложении по модулю 2 значения этого бита с теми битами контрольного кода, которые входят в ту же контрольную группу, что и обрабатываемый бит. Структурная схема тестовой системы приведена на рис. 9.

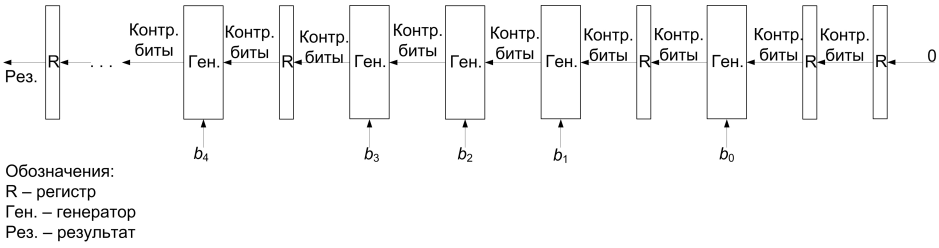

Рис. 9. Структурная схема тестовой системы на основе кода Хэмминга

На схеме показано чередование обрабатывающих стадий (генератор) и необрабатывающих (регистр). На каждой из *обрабатывающих стадий* производится сложение по модулю 2 очередного бита входных данных (b_i) с одним или несколькими контрольными битами. Номера контрольных битов, участвующие в сложении, зависят от номера стадии. Например, в соответствии с алгоритмом вычисления кода Хэмминга для обнаружения двух ошибок и исправления одной ошибки на 3-й стадии в сложении по модулю 2 будут участвовать 1-й и 2-й биты контрольного кода, на 5-й стадии — 1-й и 3-й и т. д. [7].

По формуле (4) r_1, ..., r_k — контрольные биты. В таком случае по алгоритму вычисления кода Хэмминга, чтобы определить *j*-й контрольный бит, необходимо сложить по модулю 2 все информационные биты, относящиеся к *j*-й контрольной группе. Так как операция сложения по модулю 2 обладает свойством ассоциативности, то

$$r_j = r_{jt(k-1)} \oplus b_{kj} = r_{jt(k-2)} \oplus b_{(k-1)j} \oplus b_{kj} = \\ = r_{jt1} \oplus b_{2j} \oplus \ldots \oplus b_{(k-1)j} \oplus b_{kj}, \qquad (8)$$

где r_{jt1}, ..., $r_{jt(k-1)}$ — промежуточные результаты сложения по модулю 2 информационных битов *j*-й контрольной группы на стадиях 1, …, $k-1$ конвейера, а b_{2j}, ..., $b_{(k-1)j}$ — биты *j*-й контрольной группы. В таком случае на каждой обрабатывающей ступени конвейера можно вычислить очередное значение r_{jt}.

Пусть R — набор контрольных битов на данной ступени. Тогда $R=(r_1,r_2,\ ...,\ r_k)$, где k — количество контрольных битов (контрольных групп). Фактически R_i — это набор контрольных битов для входного потока длиной i $\left(i=\overline{1,k}\right)$.

Как было показано ранее, благодаря свойству ассоциативности операции «сложение по модулю 2» каждый следующий R может быть вычислен на основе предыдущих значений. В таком случае для каждого набора контрольных битов R_i на i-й стадии конвейера справедливо выражение

$$R_i = H\left(R_{i-1}, i, b_i\right), \tag{9}$$

где H — функция, вычисляющая очередной набор контрольных битов кода Хэмминга; i — номер стадии; b_i — i-й бит входного потока. Тогда задача преобразования алгоритма вычисления кода Хэмминга сводится к получению функции $H\left(R_{i-1}, i, b_i\right)$. По алгоритму вычисления кода Хэмминга каждый контрольный бит является результатом сложения по модулю 2 всех информационных битов соответствующей контрольной группы. Тогда исходя из (4) и (9) можно сделать вывод, что

$$r_{jl} = h\left(r_{j(l-1)}, l, b_i\right), \tag{10}$$

т. е. контрольный бит j-й контрольной группы на l-й ступени зависит от значения контрольного бита j-й контрольной группы на $(l-1)$-й ступени, номера ступени и соответствующего бита входного потока.

Так как в вычислении очередного значения r_{jt} принимают участие только биты из контрольной группы j, то на обрабатывающей стадии l

$$r_{jtl} = \begin{cases} r_{jt(l-1)}, & l \notin J, \\ r_{jt(l-1)} \oplus b_l, & l \in J, \end{cases} \tag{11}$$

где J — множество элементов j-й контрольной группы. По алгоритму кода Хэмминга очередной бит принадлежит к j-й контрольной группе в том случае, если в двоичной записи номера этого бита в коде Хэмминга j-й бит равен «1». Соответственно, при вычислении r_{jt} будем проверять на равенство «1» j-го бита двоичной записи номера текущей стадии.

Необрабатывающая стадия обеспечивает корректность вычисления контрольного кода и не выполняет каких-либо вычислений.

Необрабатывающие стадии имеют номера: 1, 2, 4, 8, …, 2^m. Можно сказать, что эти стадии имитируют введение в тестовую систему очередного контрольного разряда. Для необрабатывающих стадий

$$r_{jtl} = \begin{cases} 0, \ l \in J, \ l = J[0], \\ r_{jt(l-1)}, \ l \notin J[0], \end{cases} \tag{12}$$

т. е. на этой стадии начинается вычисление очередной контрольной группы и ранее биты из этой контрольной группы в вычислении не участвовали. Начальное значение контрольного бита по алгоритму Хэмминга равно «0».

Чтобы обеспечить выявление двойных сбоев, вводится дополнительный контрольный бит — «бит двойного контроля». Бит двойного контроля (ДК) складывается по модулю 2 с очередным входным значением на каждой из обрабатывающих стадий.

Структурная схема одной ступени конвейера (обрабатывающей) приведена на рис. 10.

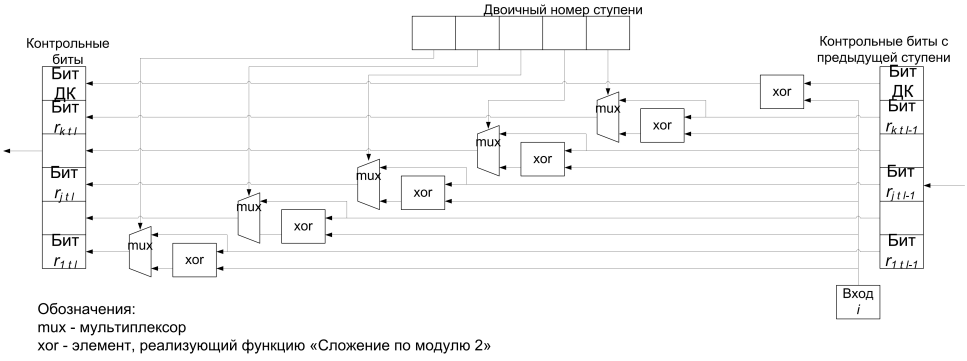

Рис. 10. Структурная схема ступени тестовой системы на основе кода Хэмминга

Номер текущей ступени является константой. Бит ДК (бит двойного контроля) представляет собой бит четности для всего полученного кода Хэмминга.

Приведем **пример**. Пусть задан входной поток 10111001. В таком случае выходными результатами стадий являются значения, приведенные в столбце 3 табл. 2 (контрольные биты до сбоя). В процессе тестирования произошел сбой, в результате которого изменилось значение 3-го бита входного потока с «0» на «1». Значения, установившиеся на выходе конвейера в результате сбоя, приведены в столбце 5 (контрольные биты после сбоя). После того как закончилась обработка искаженного входного потока, биты на выходе стадий приняли исходные значения, что отражено в столбце 7 (контрольные

биты после окончания сбоя). Далее приведем пример расчета значений r_{jt} (контрольных битов на выходе каждой ступени).

Ступень 1. Номер ступени имеет одну единицу в двоичной записи, ступень является необрабатывающей, первый контрольный бит получает значение «0». Значения остальных контрольных битов на этой стадии не определены. Для приближения к аппаратной реализации будем считать, что их значения также равны «0».

Ступень 2. Аналогично ступени 1 это необрабатывающая ступень, второй контрольный бит получает значение «0». Значение первого контрольного бита переписывается. Остальные контрольные биты также равны «0».

Ступень 3. Первая обрабатывающая ступень. Значение входного бита равно «1». Двоичный номер ступени 00011, т. е. для вычисления очередного значения 1-го и 2-го контрольных битов необходимо сложить их предыдущее значение и входной бит по модулю 2:

$$r_{1t3} = 0 \oplus 1 = 1$$

$$r_{2t3} = 0 \oplus 1 = 1$$

По договоренности бит ДК здесь — сложение по модулю 2 всех информационных битов:

$$r_{\text{ДК}\,t3} = 0 \oplus 1 = 1$$

Обрабатываемый бит не входит в 3-ю и 4-ю контрольные группы, поэтому значения 3-го и 4-го контрольных битов переписываются.

Ступень 4. Необрабатывающая ступень. Вводится 3-й контрольный бит. Он получает значение «0». Значения остальных битов переписываются.

Ступень 5. Вторая обрабатывающая ступень. Номер ступени: 00101. Значение входного бита: «0». На этой ступени вычисляются значения 1-го и 3-го контрольных битов, остальные контрольные биты переписываются:

$$r_{1t5} = 1 \oplus 0 = 1$$

$$r_{2t5} = 1 = 1$$

$$r_{3t5} = 0 \oplus 0 = 0$$

$$r_{4t5} = 0 = 0$$

$$r_{\text{ДК}\,t5} = 1 \oplus 0 = 1$$

Выполнив аналогичные вычисления для стадий 6–12, а также для измененного входного потока (имитация сбоя) и потока после окончания сбоя, получим значения, приведенные в табл. 2.

Таблица 2

Промежуточные значения контрольных битов на выходе каждой из ступеней конвейера

Номер ступени	Биты до сбоя		Биты после сбоя		Биты после окончания сбоя	
	входные	контрольные	входные	контрольные	входные	контрольные
1	2	3	4	5	6	7
$1_{10} = (00001)_2$	—	00000	—	00000	—	00000
$2_{10} = (00010)_2$	—	00000	—	00000	—	00000
$3_{10} = (00011)_2$	1	10011	1	10011	1	10011
$4_{10} = (00100)_2$	—	10011	—	10011	—	10011
$5_{10} = (00101)_2$	0	10011	0	10011	0	10011
$6_{10} = (00110)_2$	0	10011	1	00101	0	10011
$7_{10} = (00111)_2$	1	00100	1	10010	1	00100
$8_{10} = (01000)_2$	—	00100	—	10010	—	00100
$9_{10} = (01001)_2$	1	11101	1	01011	1	11101
$10_{10} = (01010)_2$	1	00111	1	10001	1	00111
$11_{10} = (01011)_2$	0	00111	0	10001	0	00111
$12_{10} = (01100)_2$	1	11011	1	01101	1	11011

Примечание: бит двойного контроля, входящий в состав контрольных битов, считается для всех битов фактического входного потока без учета самих контрольных битов. Вычисление бита ДК с учетом контрольных битов на каждой стадии не имеет смысла, так как на всех стадиях, кроме последней, значения контрольных битов являются промежуточными. Перед началом декодирования кода Хэмминга для получения значения бита ДК, полностью соответствующего алгоритму, необходимо сложить по модулю 2 итоговые значения контрольных кодов и полученное значение бита ДК. Таким образом, результирующее значение битов ДК для значений, приведенных в табл. 2:

- до сбоя — «0»;
- после сбоя — «1»;
- после окончания сбоя — «0».

Реализация тестовой системы на основе кода Хэмминга. Тестовая система состоит из модуля, представляющего собой отдельную ступень тестового конвейера, самого тестового конвейера (заданное количество соединенных между собой модулей), и может быть дополнена узлами для проверки внутренних подсистем ПЛИС, для дополнительной обработки значений со входов и т. п.

Исходные коды тестовой системы были написаны на языке описания и верификации аппаратного обеспечения SystemVerilog 2005 (IEEE 1800-2005).

На рис. 11 приведена временная диаграмма работы системы в процессе обработки этого сбоя при наличии константного входного потока. Моделирование проведено в симуляторе Mentor Graphics QuestaSim.

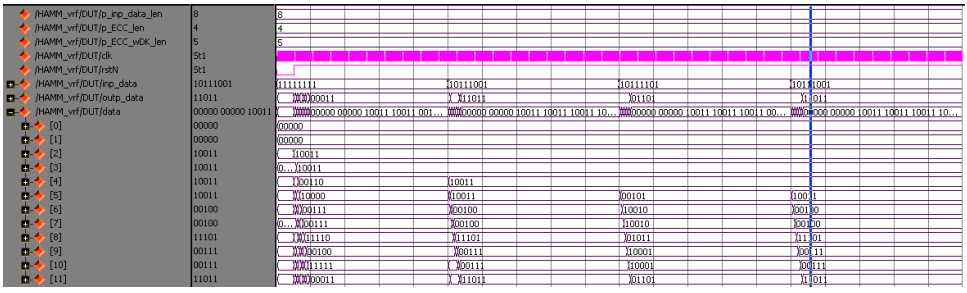

Рис. 11. Результаты моделирования системы на основе кода Хэмминга

Для вычисления места сбоя в эталонном коде Хэмминга контрольный код заменяется на код, полученный на выходе тестовой системы. Далее вычисляется синдром ошибки, по которому выявляется номер сегмента, где произошла ошибка.

В данном примере итоговый код Хэмминга, полученный в результате сбоя, имеет вид 101111001101. В результате проверки кода получаем синдром ошибки 00110, т. е. ошибка произошла в 6-м бите кода Хэмминга, что соответствует 3-му информационному биту.

В табл. 3 приведены результаты синтеза системы тестирования на основе кода Хэмминга в базис ПЛИС Actel APA 1000.

Таблица 3

Результаты синтеза конвейеризированной системы на основе кода Хэмминга

Количество ступеней	Tiles	Использовано от общего количества tiles, %	Частота, МГц
100	1629	2,89	336,36
200	3695	6,56	336,36
300	5948	10,56	336,36
400	8313	14,76	336,36
500	10 774	19,13	336,36
600	13 305	23,62	336,36
700	15 912	28,25	336,36
800	18 551	32,94	336,36
900	21 224	37,68	336,36

Аналогично конвейеризированным тестовым системам на основе CRC данная тестовая система продемонстрировала практически линейный рост количества занимаемых ресурсов при увеличении количества ступеней конвейера. Увеличение количества стадий практически не оказало влияния на максимальную частоту работы, т. е. выполняется требование масштабируемости прошивки.

Проведенное тестирование отвечает также требованиям детекции факта появления сбоя и обнаружения места его возникновения, предъявляемым к тестовой прошивке.

Заключение. Предложен и реализован метод тестирования систем хранения и обработки данных, основанных на ПЛИС, с использованием конвейеризированных генераторов контрольных кодов, позволяющих детектировать множественные ошибки и определять место возникновения сбоя. По сравнению с известными способами тестирования метод отличается большей гибкостью, большей чувствительностью к одиночным и многократным сбоям и отказам, а также возможностью выявления места возникновения сбоя и отказа.

ЛИТЕРАТУРА

[1] Зыков Д., Ковригин В. Об одном подходе к снижению трудоемкости на этапе проверки функционирования электронных устройств на ПЛИС. *Компоненты и технологии*, 2004, № 3, с. 168.

[2] Шаропин Ю., Будаев В. *Основы построения систем питания ПЛИС. Компоненты и технологии*, 2006, № 8, с. 144–151.

[3] Pratt B., Caffrey M., Graham P., Morgan K., Wirthlin M. *Improving FPGA Design Robustness with Partial TMR*. IRPS, 2006, 7 p.

[4] Бобровский Д.В. *Методы и средства прогнозирования стойкости ПЛИС к воздействию радиационных факторов космического пространства*. Дис. … канд. техн. наук. Москва, 2011, 100 с.

[5] Williams R.N. *A Painless Guide to CRC Error Detection algorithms*. Rocksoft™ Pty Ltd., Australia, 1993, 30 p.

[6] *IEEE Standard for SystemVerilog — Unified Hardware Design, Specification, and Verification Language.* The Institute of Electrical and Electronics Engineers, Inc. New York, 2005, 664 p.

[7] Никитин Г.И. Поддубный С.С. *Помехоустойчивые циклические коды*. Санкт-Петербург, СПбГУАП, 1998, 72 с.

Статья поступила в редакцию 28.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Брехов О.М., Ратников М.О. Тестирование ПЛИС с помощью конвейеризированных генераторов контрольных кодов. *Инженерный журнал: наука и инновации,* 2013, вып. 11. URL: http://engjournal.ru/catalog/it/hidden/1005.html

Брехов Олег Михайлович — д-р техн. наук, профессор, заведующий кафедрой Московского авиационного института (национального исследовательского университета). e-mail: obrekhov@mail.ru

Ратников Максим Олегович — аспирант Московского авиационного института (национального исследовательского университета). e-mail: m.o.ratnikov@mail.ru