ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2019 Управление, вычислительная техника и информатика № 47

ПРОЕКТИРОВАНИЕ И ДИАГНОСТИКА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

УДК 519.2
DOI: 10.17223/19988605/47/13

А.В. Лапутенко, Е.М. Винарский

СИНТЕЗ ТЕСТОВ ДЛЯ ЦИФРОВЫХ СИСТЕМ НА ВЫСОКОМ И НИЗКОМ УРОВНЯХ АБСТРАКЦИИ

Работа выполнена при финансовой поддержке фонда РНФ, грант № 16-49-03012.

Исследуется задача построения тестов с гарантированной полнотой для последовательностных цифровых систем. Рассматриваются тесты, нацеленные как на обнаружение в логических схемах низкоуровневых неисправностей различных типов, так и на более высокий функциональный уровень. Для построения тестов по модели логической схемы используется мутационный подход. При построении тестов с гарантированной полнотой для автоматной модели, т.е. модели более высокого уровня, тесты, построенные по логической схеме, дополняются последовательностями, которые позволяют покрыть ранее не покрытые переходы соответствующего конечного автомата. Расширенные тесты являются полными как относительно мутантов логической схемы, так и относительно автоматных ошибок. Приводятся результаты компьютерных экспериментов с описаниями последовательностных логических схем и соответствующими им конечными автоматами, подтверждающие эффективность предложенного подхода.

Ключевые слова: последовательностная цифровая система; конечный автомат; тестовые последовательности; мутационный подход; обход графа переходов.

Уровень производства цифровых устройств и использования их в различных сферах человеческой деятельности постоянно растет. Соответственно, задача эффективной проверки правильности функционирования подобных устройств на различных этапах не теряет своей актуальности, особенно для критических систем. Существует несколько подходов к построению верификационных тестов для цифровых устройств на различных уровнях абстракции [1, 2] с использованием различных моделей. Одной из наиболее ранних моделей является модель логической схемы, константные неисправности в которой достаточно полно описывали неисправности в релейно-контактных схемах [3]; тесты, построенные на основе этой модели, активно использовались, в том числе для проверки качества БИС [2, 4].

Однако почти сразу было отмечено, что тесты, построенные для обнаружения константных неисправностей логической схемы, не являются достаточно качественными при тестировании СБИС и ПЛИС, и множество константных неисправной постоянно расширялось другими моделями неисправностей [5]. Как показывают результаты экспериментов, проведенных различными исследователями (см., напр.: [6]), тесты, построенные по логическим схемам, покрывая при символьном тестировании достаточно большое количество простых путей в расширенном автомате, тем не менее оставляют непокрытыми, т.е. непроверенными, достаточно большое количество различных переходов в исходной логической схеме. С другой стороны, построение теста обходом графа переходов не расширенного, а конечного автомата, построенного по исходной логической схеме, может потребовать построения слишком большого автомата (state explosion problem). Поэтому в настоящей работе мы предлагаем так называемый гибридный подход к построению проверяющего теста для цифровой схемы. На первом шаге по заданной логической схеме строится тест, обнаруживающий три типа неисправностей [5].

На втором шаге, если по логической схеме можно построить автомат, тест достраивается до обхода графа переходов построенного автомата или любого другого автоматного теста с гарантированной полнотой; как известно, такой тест обнаруживает большое количество функциональных ошибок в реализации системы [7]. Если построение конечного автомата по логической схеме не представляется возможным, то устанавливается модель неисправности, относительно которой построенный тест является полным.

Логические схемы являются одной из наиболее популярных моделей для проектирования и анализа цифровых устройств. Распространенным способом проектирования логических схем является создание описания схемы на одном из языков логического проектирования (Hardware Description Language, HDL). Примерами таких языков являются языки Verilog, VHDL [8], BLIF, BENCH. При построении тестов для логических схем успешно применяется мутационный подход. Для имеющейся эталонной схемы рассматривается область неисправности, состоящая из описаний эталонной логической схемы с внесенными в нее неисправностями определенных типов. Распространенными типами неисправностей являются одиночные константные неисправности, проявляющиеся в том, что выход одного из вентилей схемы «залипает» на значение логического нуля или единицы. Также используется модель неисправности, содержащая неисправности типа перемычек, возникающие при ошибочном соединении выхода одного логического элемента со входом другого. Третьим типом неисправностей, рассматриваемых нами, являются трудно обнаружимые неисправности, которые появляются при замене одного вентиля схемы другим.

В предыдущих работах [5] было показано, что тесты, направленные на обнаружение одного вида неисправностей, имеют достаточно невысокий процент обнаружения неисправностей двух других типов. Из этого следует, что для построения полных тестов относительно трех перечисленных моделей неисправностей необходимо строить множество тестовых последовательностей, обнаруживающих каждую неисправную схему. Полный мутационный тест получается объединением всех таких последовательностей, возможно, с последующей минимизацией теста, т.е. удалением последовательностей, обнаруживающих одно и то же множество мутантов.

Другим подходом к построению тестов для логических схем является рассмотрение поведенческой модели более высокого уровня, функционально соответствующей данной логической схеме, и применение известных методов синтеза тестов по данной модели. В работе [6] в качестве такой модели рассматривается расширенный автомат, а качество теста оценивается покрытием простых путей в расширенном автомате. В этом случае тесты, построенные по логической схеме расширенного автомата, оказываются достаточно качественными, однако критерий покрытия путей не является достаточным для обнаружения многих функциональных ошибок. Для построения тестов, обнаруживающих большее количество функциональных ошибок, в качестве модели высокого уровня можно использовать конечный автомат. Как показано в наших предыдущих работах [9], тесты, построенные на высоком уровне абстракции (по модели конечного автомата), имеют достаточно высокий процент покрытия относительно трех типов неисправностей в логических схемах, но, тем не менее, не достигают 100%.

В данной работе предлагается подход для построения тестов для последовательностных схем, которые бы являлись полными относительно неисправностей трех рассматриваемых типов логической схемы, а также относительно выходных ошибок в конечном автомате, соответствующем данной логической схеме. Если конечный автомат, построенный по логической схеме, оказывается слишком большим, то для построенного теста устанавливается, относительно какой автоматной модели неисправности тест является полным. Таким образом, тесты, построенные предложенным методом, являются достаточно качественными для проверки отсутствия в цифровых системах функциональных ошибок как на низком, так и на высоком уровнях абстракции.

1. Основные определения

Данная работа является продолжением работ [5, 7, 9, 10], и, соответственно, мы используем основные определения из этих работ.

Под *конечным автоматом* понимается инициальный детерминированный автомат **S**, т.е. (S, I, O, $next\_state_s$, out_s, s_0), где S – конечное непустое множество состояний с выделенным начальным состоянием s_0; I и O – конечные входные и выходные алфавиты; $next\_state_s(S, I)$ – функция переходов автомата **S**, отображающая множество $S \times I$ в множество состояний S, $out_s(S, I)$ – функция выходов автомата **S**, отображающая множество $S \times I$ в выходной алфавит O. В текущем состоянии автомат принимает на вход входной символ, выдает соответствующий выходной символ и переходит в следующее состояние.

Комбинационная схема состоит из логических элементов, реализующих соответствующую булеву функцию. *Последовательностные схемы* (схемы с памятью) включают как комбинационную часть (комбинационную схему), так и элементы задержки (триггеры).

Конечный автомат, моделирующий поведение логической схемы, можно построить путем моделирования выходных реакций схемы во всех состояниях в ответ на входные воздействия. В данном случае множество S состояний конечного автомата состоит из булевых векторов (при необходимости хешированных целыми числами), представляющих комбинации значений элементов задержки, достижимых из начальной комбинации (начального состояния). Соответственно множество I входных символов автомата содержит булевы векторы, которые могут быть поданы на входы схемы.

В качестве примера на рис. 1 приведена последовательностная схема из работы [7]; конечный автомат, моделирующий ее поведение, приведен на рис. 2.

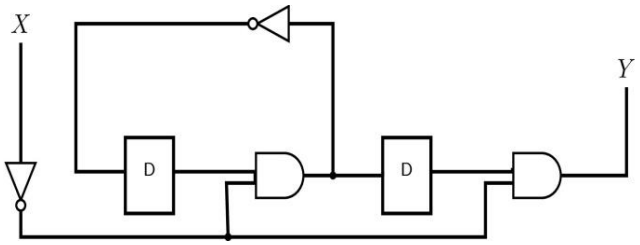

Рис. 1. Пример последовательностной схемы
Fig 1. An example of a sequential circuit

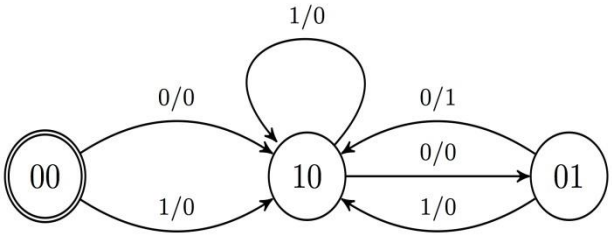

Рис. 2. Конечный автомат, описывающий поведение схемы на рис. 1
Fig 2. An FSM describing the circuit at the Fig. 1

Логическая схема имеет один вход, два элемента задержки и один выход. Конечный автомат, моделирующий ее поведение, имеет два входных символа, два выходных символа и три состояния. Начальное состояние каждого элемента задержки устанавливается в 0, поэтому начальное состояние автомата равно (00). Заметим, что состояние (11) недостижимо из начального состояния, следовательно, оно не входит в множество S = {(00), (01), (10)} состояний автомата.

Подобно работам [5, 7, 9, 10] мы рассматриваем следующие типы неисправностей, вносимых в цифровые схемы:

– *схема с одиночной константной неисправностью*, когда выход одного из вентилей схемы «залипает» на логическое значение «0» или «1»;

– *схема с одиночной неисправностью перемычки*, когда выход одного из логических элементов ошибочно соединяется с входом другого элемента;

– *схема с трудно обнаружимой неисправностью*, когда один из вентилей заменяется на вентиль, реализующий другую булеву функцию.

Проверяющий тест, полный относительно множества описанных выше неисправностей, является достаточно качественным [2, 4] и обнаруживает большое количество различных функциональных ошибок в тестируемой цифровой системе. Однако, как показывают проведенные нами эксперименты, этот тест «проходит» в среднем только по половине переходов соответствующего конечного автомата, т.е. остальные переходы системы остаются непроверенными.

2. Модели неисправности и полные проверяющие тесты

Для конечных автоматов существуют различные методы синтеза тестов с гарантированной полнотой. Однако практически в каждом методе присутствует обход графа переходов автомата. Если число элементов задержки и / или внешних входов логической схемы не меньше 20 [9], то построить соответствующий автомат и его граф переходов достаточно затруднительно. В этом случае можно построить проверяющий тест по логической схеме с использованием мутантов, описанных выше, и оценить полноту построенного теста для конечного автомата, описывающего поведение логической схемы.

Пусть поведение логической схемы описывается автоматом **S**, в котором множество состояний S есть подмножество всех двоичных наборов длины n, где n – число элементов задержки, и входной алфавит I содержит все двоичные наборы длины p, где p – число внешних входов логической схемы. Для каждой неисправности логической схемы построим соответствующий мутант; различающая последовательность для эталонной логической схемы и мутанта (если существует) строится с использованием одной из систем логического синтеза и верификации [11]. Построенные последовательности собираются в единый тест TS, который можно дополнительно оптимизировать [10]. Поведение логической схемы моделируется на последовательностях теста TS, и строится соответствующий конечный автомат $M_{LC}(TS)$.

В общем случае автомат $M_{LC}(TS)$ детерминированный и частично определенный. Построим разбиение π множества $S \times I$ на классы таким образом, что каждый класс множества содержит некоторую пару «состояние, входной символ», для которой определено поведение автомата $M_{LC}(TS)$. Рассмотрим модель неисправности $<\mathbf{S}, \cong, FD_\pi>$, где $\mathbf{S} = (S, I, O, next\_state_s, out_s, s_0)$ – полностью определенный автомат-спецификация, $\cong$ – отношение эквивалентности. Область неисправности FD_π содержит каждый полностью определенный автомат **P** с входным алфавитом I, множество P состояний которого есть подмножество множества S и в котором возможны только выходные ошибки относительно эталонного автомата **S**, причем для каждого класса b из π справедливо: либо выходной символ в автомате **P** «правильный» для всех пар класса, либо для каждой пары класса в автомате **P** есть ошибка в выходном символе.

Теорема. Проверяющий тест TS обнаруживает каждый автомат из класса FD_π, не эквивалентный эталонному автомату **S**, т.е. является полным проверяющим тестом относительно модели неисправности $<\mathbf{S}, \cong, FD_\pi>$.

Доказательство. Пусть автомат $\mathbf{P} = (P, I, O, next\_state_p, out_p, p_0) \in FD_\pi$, т.е. **P** является полностью определенным детерминированным автоматом, функция переходов которого совпадает с функцией переходов автомата-спецификации **S** для каждой пары $(p, i) \in P \times I$. Функция выходов автомата **P** может отличаться от функции выходов автомата **S**, однако для каждого класса b разбиения π множества $S \times I$ справедливо, что $\forall\ (p, i) \in b(out_p(p, i) = out_s(p, i))$ или $\forall\ (p, i) \in b(out_p(p, i) \neq out_s(p, i))$, т.е. значения функции выходов для **P** и **S** совпадают для каждой пары $(p, i) \in b$ или отличаются для всех пар $(p, i) \in b$.

Пусть в автомате **P** присутствует выходная ошибка, т.е. для некоторой пары $(p, i) \in P \times I$ имеет место $out_p(p, i) \neq out_s(p, i)$, и пара принадлежит классу b разбиения π. Тогда, по определению области неисправности, для любой пары $(p', i) \in b$ справедливо, что $out_p(p', i) \neq out_s(p', i)$. Согласно правилам

построения теста, тест *TS* содержит входную последовательность α, которая «проходит» по некоторой паре $(p', i) \in b$. По определению области неисправности FD_π, в автомате **S** рассматриваются только выходные ошибки. Соответственно, значения функции переходов в автоматах **P** и **S** совпадают для каждой пары (p, i), и последовательность α «проходит» по паре (p', i), т.е. выходные последовательности автоматов **P** и **S** на последовательность α будут различными, поскольку $out_p(p', i) \neq out_s(p', i)$. Таким образом, тест *TS* обнаружит автомат **P** с выходной ошибкой.

Следствие. Если автомат $M_{LC}(TS)$ является полностью определенным, то тест *TS* является полным относительно всех выходных неисправностей в логической схеме.

Заметим, что при необходимости тест *TS* можно достроить относительно критических переходов в достижимых состояниях, моделируя поведение логической схемы на дополнительных входных наборах. При построении полного автомата, моделирующего поведение схемы (если такое построение возможно), полученный тест является обходом графа переходов и обнаруживает большое количество функциональных ошибок на высоком (автоматном) уровне [12], а также обнаруживает и ошибки на более низком уровне, а именно на уровне логической схемы, которую далее можно реализовать в виде цифровой схемы по технологии FPGA.

В работе [9] иллюстрируется, что обход графа переходов не покрывает все ошибки, возможные в логической схеме. В следующем разделе мы иллюстрируем, что тест, построенный на основе мутантов логической схемы, не покрывает все переходы соответствующего конечного автомата, и, таким образом, объединение полученного теста с тестом, построенным обходом графа переходов автомата, дает возможность обнаружить значительно больше функциональных ошибок. Согласно проведенным экспериментам, тест, построенный по логической схеме, в среднем удлиняется примерно на 45%, если автомат имеет порядка 10 входных символов и не более 20 состояний. При увеличении размеров автомата длина теста может возрасти в 2–4 раза.

3. Описание компьютерных экспериментов

При проведении компьютерных экспериментов нами были использованы программные инструменты для генерации конечных автоматов и построения тестов обходом графа переходов [13], генерации трех множеств схем с неисправностями для эталонной схемы, представленной в BLIF описании, построении различающей последовательности для эталонной и мутированной схем [11], построении модели в виде BLIF описания по данному конечному автомату, а также скрипты на языке Perl [14] для автоматического запуска указанных инструментов и обработки результатов их работы. Часть из указанных инструментов находится в свободном доступе, часть была создана непосредственно в процессе работы.

В экспериментах использовались случайно сгенерированные автоматы с количеством состояний от 5 до 20, с числом входов от 2 до 16, а также соответствующие каждому автомату логические схемы в формате BLIF.

В первой части экспериментов было вычислено среднее значение количества мутантов логической схемы, обнаруженных тестом, построенным как обход графа переходов соответствующего конечного автомата. Результаты представлены в табл. 1.

Как видно из данной таблицы, тест, построенный обходом графа переходов соответствующего автомата, обладает достаточно высокой полнотой относительно мутаций трех типов в логических схемах, хотя полнота и не достигает 100%.

Таблица 1

Полнота конечно-автоматного теста относительно мутаций в логической схеме

Число состояний автомата	Обнаруженные мутанты соответствующей логической схемы, %
5	91,80
10	95,94
15	97,06
20	96,50

Далее были построены полные тесты относительно рассмотренных ранее видов неисправностей в логических схемах. В табл. 2 приведены средние значения длины мутационного теста для логической схемы, построенной по заданному конечному автомату.

Таблица 2

Средняя длина мутационного теста для схем, соответствующих случайно сгенерированным автоматам

Число состояний автомата	Число входных и выходных символов			
	2	4	8	16
5	19,00	26,67	37,67	83,67
10	49,33	79,00	108,67	106,00
15	46,00	55,33	101,33	114,00
20	64,67	89,00	183,00	157,33

На данном этапе экспериментов проверяющий тест строился следующим образом.

1. По случайно сгенерированному конечному автомату была построена соответствующая логическая схема в формате BLIF.

2. Для построенной логической схемы был сгенерирован тест TS_M относительно трех типов неисправностей, рассмотренных выше.

3. Поведение автомата моделировалось на тесте TS_M и определялось множество T_N переходов автомата, которые тестом не покрываются.

4. Тест TS_M дополнялся последовательностями, покрывающими переходы из множества T_N.

5. Вычислялась длина теста как сумма длин всех тестовых последовательностей.

Данная последовательность шагов была применена ко всем автоматам, построенным случайным образом на предыдущем этапе экспериментов, и в табл. 3 приведены средние значения длин тестов, полученных с использованием описанного выше алгоритма.

Таблица 3

Средняя длина теста, дополненного автоматными последовательностями

Число состояний соответствующего автомата	Число входных и выходных символов в соответствующем автомате			
	2	4	8	16
5	23,33	40,00	76,67	158,00
10	55,00	110,33	170,33	279,67
15	81,00	131,33	222,67	405,33
20	106,33	163,67	316,33	600,00

Как видно из табл. 3, для относительно небольших автоматов построенный таким образом тест в среднем имеет длину на 45% больше по сравнению с тестом, построенным при мутационном тестировании схемы, соответствующей конечному автомату, и дает возможность обнаружить значительно больше функциональных ошибок в тестируемой системе. При увеличении размеров автомата длина теста увеличивается в несколько раз, на некоторых автоматах в 2–4 раза.

Если автомат, соответствующий логической схеме, не удается построить ввиду его громоздкости, то можно либо довольствоваться полнотой теста из раздела 2, либо достраивать тест до покрытия тех переходов, которые достаточно просто промоделировать. При этом, согласно проведенным экспериментам, после увеличения длины теста в 2–4 раза можно полагать, что большое количество переходов в автомате окажется покрытым.

Заключение

Полученные экспериментальные результаты позволяют заключить, что возможно эффективное использование мутационного подхода при синтезе качественных тестов для последовательностных схем: тест, построенный относительно мутаций логической схемы, дополняется последовательностями, покрывающими непокрытые переходы соответствующего конечного автомата, пока длина исходного теста не увеличится в 2–4 раза. Согласно результатам экспериментов, можно ожидать, что тест, расширенный подобным образом, будет полным относительно большого количества ошибок на

функциональном уровне и, соответственно, тесты, построенные с помощью такого подхода, могут быть использованы в качестве проверки правильности функционирования систем как на высоком, так и на низком уровнях абстракции.

Авторы выражают благодарность доценту ТГУ М.Л. Громову за предоставленные программные инструменты, использованные в данной работе, а также постдоку университета Телеком Южный Париж Хорхе Лопезу и профессору ТГУ Н.В. Евтушенко за ценные дискуссии при подготовке данной работы.

ЛИТЕРАТУРА

1. Vasilevskii M. Failure Diagnosis of Automata // Kibernetika. 1973. V. 4. P. 98–108.
2. Пархоменко П.П., Согомонян Е.С. Основы технической диагностики. М. м: Энергоиздат, 1981. 319 с.
3. Яблонский С.В. Некоторые вопросы надежности и контроля управляющих систем // Математические вопросы кибернетики. М. : Наука, 1988. Вып. 1. С. 5–25.
4. Матросова А.Ю., Останин С.А., Бухаров А.В., Кириенко И.Е. Поиск всех тестовых наборов для неисправности логической схемы и представление их ROBDD-графом // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2014. № 2 (27). С. 82–89.
5. Кушик Н.Г., Лопез Х.Е., Евтушенко Н.В. Исследование корреляции тестовых последовательностей при проверке надежности функционирования цифровых компонентов физических систем // Известия вузов. Физика. 2016. Т. 59, № 8. С. 134–139.
6. Smolov S.A., López J., Kushik N., Yevtushenko N., Chupilko M.M., Kamkin A.S. Testing logic circuits at different abstraction levels: an experimental evaluation // Proc. of the IEEE East-West Design & Test Symposium. EWDTS. Yerevan, 2016. P. 189–192.
7. Vinarskii E., Laputenko A., López J., Kushik N. Testing Digital Circuits: Studying the Increment of the Number of States and Estimating the Fault Coverage // EDM 2018 : Proc. of the 19th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices, 2018. P. 220–224.
8. Бибило П.Н. Основы языка VHDL : учеб. пособие. М. : ЛИБРОКОМ, 2016. 328 с.
9. Lopez J., Vinarsky E., Laputenko A. On the Fault Coverage of High-level Test Derivation Methods for Digital Circuits // EDM 2017 : Proc. of the 18th International Conference on Micro/Nanotechnologies and Electron Devices, 2017. P. 184–189.
10. Лапутенко А.В., Лопез Х.Е., Евтушенко Н.В. Обработка экспериментальных данных при верификации компонентов физических систем: оценка качества тестовых последовательностей // Известия вузов. Физика. 2017. Т. 60, № 11. С. 146–151.
11. Brayton R., Mishchenko A. ABC: An Academic Industrial-Strength Verification Tool // Computer Aided Verification : Proc. of the 22nd International Conference, 2010. P. 24–40.
12. Прокопенко С.А. Минимизация проверяющих тестов для систем логического управления методами теории конечных автоматов : дис. … канд. техн. наук. Томск, 2000. 117 с.
13. Shabaldina N., Gromov M. FSMTest-1.0: a manual for researches // Proc. of IEEE East-West Design & Test Symposium (EWDTS'2015). Batumi. Georgia. September 26–29, 2015. P. 216–219.
14. Wall L., Christiansen T., Orwant J. Programming Perl. 3nd ed. Sebastopol, CA : O'Reilly & Associates, Inc, 2000. 1104 p.

Поступила в редакцию 27 сентября 2018 г.

Laputenko A.V., Vinarskii E.M. (2019) DERIVING TESTS FOR DIGITAL CIRCUITS AT LOWER AND HIGHER ABSTRACTION LEVELS. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 47. pp. 110–117

DOI: 10.17223/19988605/47/13

In this paper, we propose an approach for deriving complete test suites for logic circuits, which detect both low-level faults and high-level faults. For deriving low level tests, we consider logic circuits described in BLIF format and a mutation approach is used for the test derivation. First, logic circuit mutants with respect to considered faults are derived and test sequences, for distinguishing the reference and mutated circuits are then constructed (if such a sequence exists). For deriving a distinguishing sequence, a logic synthesis and verification software tool called ABC is used. Three types of faults in logic circuits are considered: those are Single Stuck-At Faults (SSF) considered in many references. We also consider Single Bridge Faults (SBF), which occur when one of the intermediate inputs becomes wrongly wired. The last considered faults type is a Hardly Detectable Faults (HDF) when a single gate of the circuit is replaced with another gate that has slightly different behavior. Test suite is derived by covering all such mutants. For deriving higher level tests, we use a finite state machine (FSM) and according to our experiments test suites derived based on logic circuits cover 64% of transitions of the corresponding FSM on average. We propose to complete logic circuit based test suites with sequences, which allow to cover previously uncovered transitions of the corresponding FSM and specify the fault model for which augmented test suites are complete. Correspondingly, augmented tests are complete with respect to the mutations of a logic circuit as well as with respect to output faults of a higher-level FSM model.

Keywords: sequential circuits; finite state machines; test sequences; mutation approach; transition tour.

LAPUTENKO Andrey Vladimirovich (Post-graduate Student, National Research Tomsk State University, Tomsk, Russian Federation).
E-mail: laputenko.av@gmail.com

VINARSKII Evgenii Maximovich (Student, Lomonosov Moscow State University, Moscow, Russian Federation).
E-mail: vinevg2015@gmail.com

REFERENCES

1. Vasilevskii, M. (1973) Failure Diagnosis of Automata. *Kibernetika*. 4. pp. 98–108.
2. Parkhomenko, P.P. & Sogomonyan, E.S. (1981) *Osnovy tekhnicheskoy diagnostiki* [The basic of technical diagnostics]. Moscow: Energoizdat.
3. Yablonskiy, S.V. (1988) Nekotorye voprosy nadezhnosti i kontrolya upravlyayushchikh sistem [Some problems of control systems reliability and management]. In: Yablonskiy, S.V. (ed.) *Matematicheskie voprosy kibernetiki* [Mathematical problems of cybernetics]. Vol. 1. Moscow: Nauka. pp. 5–25.
4. Matrosova, A.Yu., Ostanin, S.A., Bukharov, A.V. & Kirienko, I.E. (2014) Generating all test patterns for a given stuck-at fault of a logical circuit and its ROBDD implementation. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vy-chislitel'naya tekhnika i informatika – Tomsk State University Journal of Control and Computer Science*. 2(27). pp. 82–89. (In Russian).
5. Kushik, N.G., López, J.E. & Evtushenko, N.V. (2016) Studying the correlation of test sequences in checking reliability of digital components of physical systems. *Russian Physics Journal*. 59(12). pp. 134–139.
6. Smolov, S.A., López, J., Kushik, N., Yevtushenko, N., Chupilko, M.M. & Kamkin, A.S. (2016) Testing logic circuits at different abstraction levels: an experimental evaluation. *Proceedings of the IEEE East-West Design & Test Symposium. EWDTS*. Yerevan. pp. 189–192. DOI: 10.1109/EWDTS.2016.7807687
7. Vinarskii, E., Laputenko, A., López, J. & Kushik, N. (2018) Testing Digital Circuits: Studying the Increment of the Number of States and Estimating the Fault Coverage. *EDM 2018*. Proc. of the 19th International Conference of Young Specialists on Micro / Nanotech-nologies and Electron Devices. pp. 220–224. DOI: 10.1109/EDM.2018.8435051
8. Bibilo, P.N. (2016) *Osnovy yazyka VHDL* [The basics of VHDL language]. Moscow: LIBROKOM.
9. Lopez, J., Vinarskii, E. & Laputenko, A. (2017) On the Fault Coverage of High-level Test Derivation Methods for Digital Circuits. *EDM 2017*. Proc. of the 18th International Conference on Micro/Nanotechnologies and Electron Devices. pp. 184–189. DOI: 10.1109/EDM.2017.7981736
10. Laputenko, A.V., Lopez, J.E. & Evtushenko, N.V. (2017) Processing of experimental data in the verification of physical system components: evaluation of the quality of test sequences. *Russian Physics Journal*. 60(11). pp. 146–151.
11. Brayton, R. & Mishchenko, A. (2010) ABC: An Academic Industrial-Strength Verification Tool. *Computer Aided Verification*. Proc. of the 22nd International Conference. pp. 24–40.
12. Prokopenko, S.A. (2000) *Minimizatsiya proveryayushchikh testov dlya sistem logicheskogo upravleniya metodami teorii konechnykh avtomatov* [Minimization of verification tests for logic control systems using finite state machine methods]. Engineering Cand. Diss. Tomsk.
13. Shabaldina, N. & Gromov, M. (2015) FSMTest-1.0: a manual for researches. *Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2015)*. Batumi. Georgia. September 26–29, 2015. pp. 216–219.
14. Wall, L., Christiansen, T. & Orwant, J. (2000) *Programming Perl*. 3nd ed. Sebastopol, CA: O'Reilly & Associates, Inc.