УДК 004.451.7 DOI: 10.24151/1561-5405-2020-25-4-310-319

Формализация выбора отладочных тестов при проектировании цифровых микроэлектронных систем на основе проверки выполнения требуемых функций

А.Д. Иванников, А.Л. Стемпковский

Институт проблем проектирования в микроэлектронике Российской академии наук, г. Москва, Россия

adi@ippm.ru

Для отладки проектов цифровых микроэлектронных систем необходимо формирование некоторого набора тестовых воздействий на моделируемую систему с целью проверки правильности ее функционирования. Для большого количества цифровых систем характерно выполнение последовательности функций из конечного алфавита. В работе определена частичная полугруппа на множестве допустимых последовательностей функций. Допустимые последовательности формализованы путем введения графа функций, задающего возможные для выполнения функции для различных состояний цифровой системы. Граф функций совместно с множествами входных взаимодействий для каждой функции задают спецификацию внешнего поведения цифровой системы Установлено, что если допустимость последовательного выполнения двух функций зависит от выполненных ранее функций и состояния цифровой системы, то некоторые функции должны быть разделены на подфункции. Показано, что набор отладочных тестов должен включать в себя как проверку выполнения последовательностей функций, так и правильность выполнения каждой функции с различными наборами параметров

Ключевые слова: отладка методом моделирования; цифровые системы; алфавит функций; графовое представление последовательности функций

Для цитирования: Иванников А.Д., Стемпковский А.Л. Формализация выбора отладочных тестов при проектировании цифровых микроэлектронных систем на основе проверки выполнения требуемых функций // Изв. вузов. Электроника. 2020. Т. 25. № 4. С. 310–319. DOI: 10.24151/1561-5405-2020-25-4-310-319

Formal Choice of Debugging Tests for Digital Microelectronic Systems Designs Based on the Required Functions Fulfillment Checking

A.D. Ivannikov, A.L. Stempkovskiy

Institute for Design Problems in Microelectronics of Russian Academy of Sciences, Moscow, Russia

adi@ippm.ru

Abstract: For digital microelectronic system design debugging, it is necessary to form a certain set of test influences on the simulated system to verify the correctness of its functioning. For a large number of digital systems, a sequence of functions from a finite alphabet is characteristic. It is shown that a partial semigroup is defined on the set of admissible sequences of functions. Valid sequences are formalized by introducing a graph of functions that defines the functions that can be performed for various states of the digital system. The function graph, together with the sets of input interactions for each function, specifies the specification of the external behavior of the digital system. If the admissibility of the sequential execution of two functions depends on previously performed functions and the state of the digital system, then some functions should be divided into subfunctions. It has been shown that a set of debugging tests should include both checking the execution of sequences of functions and the correctness of each function with various sets of parameters

Keywords: design debugging by simulation; digital systems; function alphabet; graph representation of a sequence of functions

***For citation*:** Ivannikov A.D., Stempkovskiy A.L. Formal choice of debugging tests for digital microelectronic systems designs based on the required functions fulfillment checking. *Proc. Univ. Electronics*, 2020, vol. 25, no. 4, pp. 310–319. DOI: 10.24151/1561-5405-2020-25-4-310-319

Введение. При проектировании цифровых микроэлектронных систем для отладки проектов широко используется метод моделирования [1–6]. На компьютерную модель цифровой системы подаются некоторые входные воздействия, а реакция модели проектируемой системы проверяется на соответствие техническому заданию. Выполнение какой-либо функции может быть инициировано не только входным сигналом цифровой системы управления, но и самой цифровой системой. Поэтому в качестве аргументов функционирования цифровых систем могут рассматриваться входные взаимодействия – последовательности сигналов, включающие как входные сигналы цифровой системы, так и ее выходные сигналы управления обменом [7].

Постановка задачи. Будем рассматривать логическую модель сигналов на шинах и линиях цифровых систем, т.е. считать, что значения сигналов представляются как 0 или 1 на линиях и как число из диапазона $0 – (2^n – 1)$ на шинах системы. Цифровые сигналы

внешних шин и линий назовем терминальными переменными – множеством **P**. Переменная $p \in \mathbf{P}$ всегда имеет одно из значений конечного множества $\mathbf{Z}_p$, элементы которого определяют целочисленное значение сигнала и направленность работы шины или линии.

Событием по переменной p называется изменение переменной p со значения $z_1 \in \mathbf{Z}_p$ на значение $z_2 \in \mathbf{Z}_p$ в момент времени t. Обозначим такое событие χ^{t}_{p,z_1,z_2}. Взаимодействие цифровой системы с внешней средой есть последовательность переключений сигналов на терминальных шинах и линиях, т.е. последовательность событий.

Входное взаимодействие μ может быть представлено в виде вектора $\left(z^{\text{н}}_{p_1},\ldots,z^{\text{н}}_{p_k}\right)$ начальных значений переменных $p_1,\ldots, p_k$ в момент времени $t = 0$ и последовательности событий по этим переменным с конечным числом событий за любой конечный интервал времени:

$$\mu=\left(z^{\text{н}}_{p_1},\ldots,z^{\text{н}}_{p_n+q}\right),\chi^{t_1}_{p_{i_1},z_{j_1},z_{j_2}},\chi^{t_2}_{p_{i_2},z_{j_3},z_{j_4}},\chi^{t_3}_{p_{i_3},z_{j_5},z_{j_6}}\ldots, \quad (1)$$

где $p_{i_1}, p_{i_2}, p_{i_3},\ldots$ – переменные, принадлежащие множеству **P** и являющиеся входными переменными и выходными переменными управления обменом; $z_{j_1}, z_{j_3}, z_{j_5},\ldots$ – значения переменных непосредственно перед событием; $z_{j_2}, z_{j_4}, z_{j_6},\ldots$ – значения переменных непосредственно после события; $\chi^{t_1}_{p_{i_1},z_{j_1},z_{j_2}},\chi^{t_2}_{p_{i_2},z_{j_3},z_{j_4}},\chi^{t_3}_{p_{i_3},z_{j_5},z_{j_6}}\ldots$ – входные и выходные события управления обменом; $t_1 \le t_2 \le t_3 \le \ldots$ – упорядоченная последовательность времен событий входного взаимодействия.

При отладке методом моделирования важной задачей является выбор конечного числа конечных по времени тестовых входных взаимодействий (тестовых примеров). Отличительная особенность предлагаемого подхода – отсутствие необходимости иметь полную формальную спецификацию на функционирование проектируемой цифровой системы. Разработчик на основании понимания функционирования системы формулирует требуемые функции с различными классами параметров и режимами работы, объединяет или разделяет функции в соответствии со своим представлением о различных режимах их выполнения.

Функционирование многих управляющих цифровых систем может быть представлено как последовательность выполняемых функций, каждая из которых задается подачей на цифровую систему некоторого входного взаимодействия. Другими словами, цифровая система выполняет некоторую последовательность функций из конечного алфавита **K**, причем выполнение каждой функции вызывается одним из входных взаимодействий определенного класса. Входные взаимодействия, задающие выполнение цифровой системой функции $k \in \mathbf{K}$, могут различаться значениями данных, временными соотношениями между отдельными сигналами в допустимых пределах и т.д. Следует отметить, что для многих современных сложных цифровых систем представление их работы как последовательного потока выполняемых функций может оказаться затруднительным. Применение предлагаемого подхода к выбору отладочных тестов на основе проверки правильности выполнения функций возможен для ограниченных по сложности микроэлектронных цифровых систем.

Отладочные тестовые примеры должны осуществлять проверку правильности выполнения цифровой системой всех ее функций из множества **K**, а также правильности выполнения допустимых последовательностей функций $f \in \mathbf{F}$, где **F** – множество допустимых последовательностей функций из множества **K**.

Формальное представление множества последовательностей выполняемых функций. Рассматривая последовательное выполнение функций k_i и k_j как операцию умножения, а последовательность k_ik_j как произведение, можно сказать, что на множестве последовательностей выполняемых функций **F** определена полугруппа. Рассмотрим полугруппу $<\mathbf{F}, \cdot>$, в общем случае частичную, с конечным множеством **K** порождающих элементов k. При этом возможны три случая.

1. Произведение $f' \cdot f''$ определено для всех $f' \in \mathbf{F}$, $f'' \in \mathbf{F}$. В этом случае полугруппа $<\mathbf{F}, \cdot>$ является полной.

2. Произведение $f' \cdot f''$ определено, если f' и f'' могут быть представлены как $f'=k_{i_1}\cdot k_{i_2}\cdot\ldots\cdot k'$, $f''=k''\cdot k_{j_2}\cdot\ldots\cdot k_{j_i}$, причем произведение $k'\cdot k''$ существует. В этом случае вопрос о существовании $f' \cdot f''$ сводится к вопросу о существовании произведений $k' \cdot k''$, где $k' \in \mathbf{K}$, $k'' \in \mathbf{K}$.

3. Произведение $f' \cdot f''$ определено, если существуют произведения $k_{i_1}\cdot k_{i_2}\cdot\ldots\cdot k'\cdot k''$, $k_{i_1}\cdot k_{i_2}\cdot\ldots\cdot k'\cdot k''\cdot k_{j_2},\ldots,k_{i_1}\cdot k_{i_2}\cdot\ldots\cdot k'\cdot k''\cdot k_{j_2}\cdot\ldots\cdot k_{j_i}$. В этом случае вопрос о существовании $f' \cdot f''$ сводится к вопросу о существовании произведения $f \cdot k$, где $f \in \mathbf{F}$, $k \in \mathbf{K}$.

Случай 3 является наиболее общим. Проанализируем его. Проектируемые цифровые системы всегда имеют конечное множество внутренних состояний. В связи с этим все существующие произведения могут быть заданы конечным числом правил. Множество **F** есть множество слов в конечном алфавите **K**, которое может быть задано некоторой праволинейной грамматикой [8] с конечным числом правил вывода. Множество допустимых слов **F** может быть задано конечным инициальным автоматом без выходов:

$$A= (\mathbf{K}, \mathbf{X}, x_н, \varphi), \qquad (2)$$

где **K** – множество входных символов; **X** – множество состояний автомата A; $x_н$ – начальное состояние, $x_н \in \mathbf{X}$; φ:$\mathbf{K}\times\mathbf{X}\rightarrow\mathbf{X}$– частичное переходное отображение.

Автомат A задает все возможные последовательности функций, выполняемые цифровой системой, т.е. множество допустимых слов **F**. Будем называть автомат A автоматом функций. Спецификации на входные взаимодействия цифровой системы могут определяться заданием множества входных взаимодействий, соответствующих каждой функции k, $k \in \mathbf{K}$ [6], и автомата функций A.

В качестве примера рассмотрим цифровую систему управления некоторым роботом, который должен двигаться под управлением последовательности поступающих командных кадров, каждый из которых задает движение робота по отрезку прямой заданной длины и направления или дуге окружности с заданными параметрами. Движение робота осуществляется в рамках некоторого прямоугольника, за границы которого робот выходить не может, и его начальное положение задано в центре прямоугольника. Для такого простейшего случая перечень функций, выполняемых цифровой системой управления, будет включать в себя движение по отрезку прямой $k_{отр}$ и движение по дуге окружности $k_{дуг}$, причем для обеих функций существует множество значений допустимых параметров. Всегда ли будет существовать произведение $k_i \cdot k_j$, т. е. будет ли всегда допустимо выполнение k_j после k_i? Возможность выполнения функции движения по отрезку прямой или дуге окружности будет существовать, если это движение не выводит робота за пределы заданного прямоугольника, т.е. фактически зависит от начального положения робота (от предыстории выполнения предыдущих функций) и параметров функции движения.

Разделим каждую из функций $k_{отр}$ и $k_{дуг}$ на подфункции: $k'_{отр}$, $k''_{отр}$ и $k'_{дуг}$, $k''_{дуг}$, где $k'_{отр}$ и $k'_{дуг}$ – функции полного выполнения движения по отрезку прямой или дуге окружно-

сти; $k''_{\text{отр}}$ и $k''_{\text{дуг}}$ – функции движения до границы заданного прямоугольника, включая движение нулевой длины. В этом случае все произведения, где первыми и вторыми сомножителями являются $k'_{\text{отр}}$, $k''_{\text{отр}}$, $k'_{\text{дуг}}$, $k''_{\text{дуг}}$, существуют.

Таким образом, с помощью модификации множества выполняемых цифровой системой функций **K**, наиболее общий случай 3 приведен к случаю 2, когда произведение $f' \cdot f''$ определено, если f' и f'' могут быть представлены как $f' = k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k'$, $f'' = k'' \cdot k_{j_2} \cdot \ldots \cdot k_{j_i}$, причем произведение $k' \cdot k''$ существует. Такое задание множества выполняемых функций **K′** существенно облегчает задание множества отладочных тестов. Далее будем рассматривать модифицированный алфавит функций **K′** как алфавит **K**. В связи с тем что в цифровых системах с конечным алфавитом выполняемых функций множество внутренних состояний всегда конечно, преобразование начального алфавита выполняемых функций в модифицированное, где вопрос существования произведения двух функций не зависит от предыстории, всегда возможен. Тогда для каждого k_i, $k_i \in$ **K**, имеется подмножество $\mathbf{K}^i$, $\mathbf{K}^i \subseteq$ **K,** такое, что $k_i \cdot k_j$, где $k_j \in \mathbf{K}^i$, всегда существует. В этом случае на множестве слов **F** задано отношение эквивалентности o такое, что $(f_1, f_2) \in$ o тогда и только тогда, когда f_1 и f_2 кончаются на одну букву.

Отношение эквивалентности o определяет разбиение **F** на классы эквивалентности $\mathbf{F}^i$, $i = 1, 2, \ldots, n$, где n – мощность алфавита **K**. При отождествлении классов $\mathbf{F}^i$, $i = 1, 2, \ldots, n$, и $\mathbf{F}^x$, $x \in$ **X**, и соответственно множества внутренних состояний автомата (2) и множества **K** автомат A может быть заменен автоматом

$$A' = (\mathbf{K}, \mathbf{K}, x_{\text{н}}, \varphi), \tag{3}$$

где **K** – множество входных символов и множество внутренних состояний; $x_{\text{н}}$ – начальное состояние, $x_{\text{н}} \in \mathbf{K}$; $\varphi: \mathbf{K} \times \mathbf{K} \to \mathbf{K}$ – частично определенное переходное отображение. Каждое внутреннее состояние k, $k \in \mathbf{K}$, задает множество слов, заканчивающихся на букву k.

Представление автомата функций в виде (3) удобно как при составлении спецификаций на входные взаимодействия, так и при использовании для выбора отладочных тестов.

Методика формирования автомата функций. Автомат функций вида (3) вместе с множествами $\mathbf{M}^k$ входных взаимодействий, соответствующих каждой функции k, является формальной спецификацией входных взаимодействий **M**. При этом множество входных взаимодействий может быть представлено следующим образом:

$$\mathbf{M} = (\mathbf{K}, A', \{\mathbf{M}^k | k \in \mathbf{K}\}),$$

где **K** – алфавит функций; A′ – автомат функций вида (3), определяющий допустимые последовательности функций **F**; $\mathbf{M}^k$ – множество входных взаимодействий, соответствующих k-й функции цифровой системы или блока.

Методика получения автомата функций по неформальному описанию работы цифровой системы или блока, приводимому в ТЗ, состоит в следующем.

1. Составить начальный вариант перечня выполняемых системой функций и описать соответствующие входные взаимодействия.

2. Выделить пары функций, возможность последовательного выполнения которых зависит от предыстории работы цифровой системы.

3. Разбить вторые элементы выделенных пар функций на подфункции либо по физическому смыслу, либо в зависимости от состояния системы, в котором инициируется второе входное взаимодействие пары, таким образом, чтобы можно было задать возможные пары функций независимо от предыстории работы.

4. Составить автомат функций A′. Одним из способов задания такого автомата функций является указание для каждой функции $k_i \in \mathbf{K}$ набора функций $\mathbf{K}_j$, каждая функция из которого может быть выполнена после k_i.

Следует отметить, что выделение набора функций цифровой системы в определенной степени процесс субъективный. Так, в случае рассмотренного робота подача управляющего кадра может быть представлена как последовательный ввод ряда чисел; ввод кадра и его отработка могут быть представлены как разные функции. Однако, несмотря на различный вид автомата функций A′ при выборе различных множеств **K**, в связи с однозначностью требований к поведению цифровой системы получаемая спецификация входных взаимодействий приведет к проверке функционирования отлаживаемой системы одинаковым образом.

Формализация подхода при выборе отладочных тестов для проверки выполнения функций. Выполнение цифровой системой функции k инициируется одним из входных взаимодействий множества $\mathbf{M}^k$. В случае если разработчик выделил достаточно крупные функции, то в зависимости от параметров функции ее выполнение может осуществляться различными блоками программного обеспечения в различных режимах работы блоков цифровой системы. В связи с этим при формировании набора отладочных тестов проекта желательно проверить правильность функционирования проекта для различных наборов параметров каждой функции k.

Представим множество значений данных $\boldsymbol{D}$ входных взаимодействий, обусловливающих выполнение цифровой системой определенной функции k, как $\boldsymbol{D}=\mathrm{U}_{i=1}^{n}\boldsymbol{D}_i$, $\boldsymbol{D}_i\cap\boldsymbol{D}_j=\varnothing$ при $i\neq j$. Каждое подмножество $\boldsymbol{D}_i$, в свою очередь, может быть разбито на непересекающиеся подмножества $\boldsymbol{D}_{ij}$ и т.д. Разбиение множества значений данных должно осуществляться исходя из физического смысла задачи таким образом, чтобы входные взаимодействия с данными различных подмножеств $\boldsymbol{D}_i$ обусловливали несколько различных алгоритмов их обработки. Самой большой группой являются множества $\mathbf{M}^k$, $k \in \mathbf{K}$, входных взаимодействий для выполнения цифровой системой функции k.

Математическим аналогом понятия «близости» входных взаимодействий является отношение эквивалентности [9]. Пусть имеется конечное множество $\boldsymbol{\Lambda}$ отношений эквивалентности λ. В частности, множество $\boldsymbol{\Lambda}$ обязательно включает в себя отношение эквивалентности λ′, λ″.

Пара входных взаимодействий эквивалентны друг другу, т.е. $(\mu_1, \mu_2) \in \lambda'$ тогда и только тогда, когда $\mu_1 \in \mathbf{M}^k$ и $\mu_2 \in \mathbf{M}^k$. Именно это отношение эквивалентности позволяет выделить входные взаимодействия, обусловливающие выполнение функции k, в множество $\mathbf{M}^k$.

Пара входных взаимодействий эквивалентны друг другу, т.е. $(\mu_1, \mu_2) \in \lambda''$, где $\mu_1 \in \mathbf{M}^k$, $\mu_2 \in \mathbf{M}^k$, тогда и только тогда, когда набор данных d_1 и d_2, присутствующих в μ_1, μ_2, принадлежит одной и той же подобласти $\boldsymbol{D}_i$ данных, где $\mathrm{U}_{i=1}^{n}\boldsymbol{D}_i=\boldsymbol{D}$, $\boldsymbol{D}_i\cap\boldsymbol{D}_j=\varnothing$ при $i\neq j$, $\boldsymbol{D}$ – область данных для $\mathbf{M}^k$. Так, возможен случай, что $(\mu_1, \mu_2) \in \lambda''$ тогда и только тогда, когда μ_1 и μ_2, заданные в виде (1), отличаются в ряде событий только значениями z'_j или z''_j, соответствующими различным данным на информационных входах, обрабатываемых по одинаковому алгоритму и изменяющими выходные последовательности цифровой системы или ее блока только в части значений на информационных выходах. Может быть задано несколько отношений эквивалентности такого типа для различных разбиений области данных $\boldsymbol{D}$.

Выбор множества $\boldsymbol{\Lambda}$ отношений эквивалентности должен осуществляться разработчиком исходя из требуемого поведения разрабатываемой цифровой системы и физического смысла задачи. При этом множество $\boldsymbol{\Lambda}$ задается косвенно.

Каждое отношение эквивалентности λ множества $\mathbf{\Lambda}$ задается на своем множестве. Так, отношение λ′ задано на всем множестве $\bar{\mathbf{M}} = \mathrm{U}_{k \in \mathbf{K}} \mathbf{M}^k$. На каждом множестве $\mathbf{M}^k$, являющемся классом эквивалентности $\bar{\mathbf{M}}$ по λ′, задаются свои отношения эквивалентности типа отношения λ″. Так, возможен случай, когда области данных для $\mathbf{M}^k$ есть $\boldsymbol{D} = \boldsymbol{D}^1 \times \boldsymbol{D}^2 \times ... \times \boldsymbol{D}^n$, а на множестве значений $\boldsymbol{D}^i$, $i = 1,\ldots, n$, каждого параметра входных взаимодействий из $\mathbf{M}^k$ определены свои отношения эквивалентности $\lambda_1, \lambda_2,\ldots,\lambda_n$, где $\lambda_1 \in \mathbf{\Lambda}$, $\lambda_2 \in \mathbf{\Lambda}$,…, $\lambda_n \in \mathbf{\Lambda}$. Возможен и другой случай, когда отношение λ_1 определяет разбиение $\boldsymbol{D} = \mathrm{U}_{i=1}^{n} \boldsymbol{D}_i$, $\boldsymbol{D}_i \cap \boldsymbol{D}_{i'} = \varnothing$ при $i \neq i'$, а отношение λ_2 определено только на одном подмножестве или на части подмножеств $\boldsymbol{D}_{i_1}, \ldots, \boldsymbol{D}_{i_l}$, где $\{i_1,\ldots,i_l\} \subset \{1,\ldots,n\}$.

Если на множестве $\mathbf{M}^k$ задано отношение λ_1, $\lambda_1 \in \mathbf{\Lambda}$, то существует разбиение $\mathbf{M}^k$ на классы эквивалентности $\mathbf{M}^{ki}$ по отношению λ_1. Если на некотором $\mathbf{M}^{ki}$ определено отношение λ_2, $\lambda_2 \in \mathbf{\Lambda}$, то для μ, где $\mu \in \mathbf{M}^{ki}$, существует произведение эквивалентностей $\lambda_1 \cdot \lambda_2$, которое всегда является эквивалентностью [9]. Произведения эквивалентностей $\prod_{\lambda_i \in \mathbf{\Lambda}'} \lambda_i$, где $\mathbf{\Lambda}' \subseteq \mathbf{\Lambda}$, которые всегда являются эквивалентностями, позволяют провести классификацию множества $\mathbf{M}^k$ с той степенью подробности, которая необходима разработчику, определившему множество $\mathbf{\Lambda}$. Входные взаимодействия каждого класса эквивалентности, определяемого максимально возможными произведениями $\prod_{\lambda_i \in \mathbf{\Lambda}'} \lambda_i$, где $\mathbf{\Lambda}' \subseteq \mathbf{\Lambda}$, различаются только значениями ряда t_j в (1) в пределах, не нарушающих ограничений на допустимые времена событий.

Как уже отмечалось, самыми крупными классами эквивалентности на множестве $\bar{\mathbf{M}}$ являются множества $\mathbf{M}^k$, соответствующие выделению алфавита выполняемых функций $\mathbf{K}$. Так, в случае упомянутого робота на множестве входных взаимодействий для движения по отрезку прямой может быть выделено отношение эквивалентности λ_1, значения признаков которого выделяют подмножества горизонтальных, вертикальных и наклонных отрезков. Если бы в качестве одной из функций цифровой системы управления роботом была выбрана функция движения, определяемая одним поступившим кадром, то тогда можно было бы ввести отношение эквивалентности λ_2, значения признаков которого выделяли бы отрезки прямой и дуги окружности.

Составление набора отладочных тестов для конкретной функции. Рассмотрим множество входных взаимодействий $\mathbf{M}_k$, инициирующих выполнение цифровой системой функции k. Несмотря на то что все входные взаимодействия множества инициируют выполнение одной и той же функции k, ее выполнение может осуществляться поразному с использованием различающихся в какой-то степени алгоритмов и, соответственно, различных режимов работы аппаратного обеспечения и различных ветвей программного обеспечения. При этом все множество входных взаимодействий может быть представлено как объединение непересекающихся подмножеств, каждое из которых инициирует выполнение специфического варианта функции k. При выборе множества отладочных тестов желательно предусмотреть проверку правильности выполнения каждого такого специфического варианта.

Разработчик для каждой функции формирует набор признаков $\zeta_1, \zeta_2, \ldots, \zeta_n$ и алфавиты их значений. Пусть имеется конечное множество признаков $\zeta_1, \zeta_2, \ldots, \zeta_n$, каждый из которых может принимать конечное множество значений $\mathbf{Z}^*_1, \mathbf{Z}^*_2, \ldots, \mathbf{Z}^*_n$. Тогда взаимосвязь признаков можно представить двудольным графом G(**V**,**E**). Множество вершин этого графа есть $\nu = \{\zeta_1,\ldots,\zeta_n\} \cup \{\mathrm{U}_{i=1}^{n} \mathrm{Z}_i^*\}$. Одна группа вершин представляет собой множество признаков, а другая группа – возможные значения каждого признака. Множество ребер $\mathbf{E} = \mathbf{E}^1 \cup \mathbf{E}^2$, $\mathbf{E}^1 \cap \mathbf{E}^2 = \varnothing$. Ребра множества $\mathbf{E}^1$ соединяют вершину ζ_i с вер-

шиной z, если $z \in Z_i^*$. Ребра множества $\mathbf{E}^2$ соединяют вершину z_i^j, где $z_i^j \in Z_i^*$, с вершиной ζ_l, если признак ζ_l определен для входного взаимодействия μ в случае, когда для этого входного взаимодействия $\zeta_i = z_i^j$.

Множество отладочных тестов считается полным, если для любого признака ζ_i, $i = 1, \ldots, n$, и $z \in Z_i^*$ в множестве отладочных тестов найдется, по крайней мере, одно входное взаимодействие, для которого $\zeta_i = z$. Минимальным полным множеством отладочных тестов назовем такое, число тестов в котором минимально. Задача составления минимального полного множества отладочных тестов для проверки выполнения каждой функции состоит в выборе сочетаний значений признаков для каждого отладочного теста множества.

В работе [10] показано, что минимальная мощность минимального полного множества отладочных тестов есть $\max|Z_i^*|$, $i = 1, \ldots, n$, где n – количество признаков $\zeta_1, \ldots, \zeta_n$; максимальная мощность минимального полного множества отладочных тестов есть $\sum_{i=1}^{n} (|Z_i^*| - 1) + 1$.

Последовательность действий для формирования множества тестовых примеров для отладки проектов цифровых систем. Рассмотрим последовательность действий для формирования множества тестовых примеров для отладки проектов цифровых систем, функционирование которых может быть представлено как последовательность выполнения функций из конечного алфавита.

1. Разработчик формирует начальный, конечный по мощности, перечень функций, выполняемых цифровой системой, а именно алфавит $\mathbf{K}$.

2. Для каждой функции $k_i \in \mathbf{K}$ разработчик определяет, какие функции $k_j \in \mathbf{K}$, $j = 1, \ldots, n$, где n – количество элементов в алфавите $\mathbf{K}$, могут выполняться после k_i. При этом для каждого k_i выделяется подмножество $\{k_{j_1}, \ldots, k_{j_m}\} \subset \{k_{i_1}, \ldots, k_{i_n}\}$, для каждого элемента k_{jl} которого существование произведения $k_i \blacksquare k_{j_l}$ зависит от некоторых условий, т.е. некоторой предыдущей последовательности функций, или от внутреннего состояния цифровой системы. Затем каждая функция k_{j_l} разбивается на подфункции $k_{j_l}^{\,1}, \ldots, k_{j_l}^{\,q}$. Если выполнение функции k_{j_l} инициируется одним из входных взаимодействий множества $\mathbf{M}_{j_l}$, то это множество разбивается на непересекающиеся подмножества $\mathbf{M}_{j_l}^r$, где $r = 1, \ldots, q$, и $\mathbf{M}_{j_l} = \mathrm{U}_{r=1}^{q} \mathbf{M}_{j_l}^r$. Таким образом формируется новый (также конечный по мощности) перечень функций, выполняемых цифровой системой, а именно модифицированный алфавит $\mathbf{K}_{\text{мод}}$. При дальнейшем описании будем рассматривать этот модифицированный алфавит как алфавит функций, выполняемый цифровой системой, и будем обозначать его $\mathbf{K}$.

3. Для полученного алфавита функций $\mathbf{K}$ и частичной полугруппы $<\mathbf{F}, \cdot>$ с конечным множеством $\mathbf{K}$ порождающих элементов k строится граф переходов $G(\mathbf{V},\mathbf{E})$ автомата функций $\mathbf{A}_\text{ф}$. В графе $G(\mathbf{V},\mathbf{E})$ каждая вершина $v \in \mathbf{V}$ помечена одним из элементов k алфавита $\mathbf{K}$, т.е. представляет все слова $\mathbf{F}^k \subset \mathbf{F}$, заканчивающиеся элементом k алфавита функций. Исключение – вершина $v_{\text{нач}}$, соответствующая пустому слову и представляющая собой состояние, в котором цифровая система оказывается после включения питания. Множество ребер $e_{ij} \in \mathbf{E}$ графа $G(\mathbf{V},\mathbf{E})$ определяет возможные произведения $k_i \blacksquare k_j$, т.е. для каждой вершины, помеченной k_i, выходят ребра в те вершины k_j, которые помечены функцией алфавита $\mathbf{K}$, которая может быть выполнена после функции k_i, т.е. условием наличия ребра e_{ij} является существование произведения $k_i \blacksquare k_j$.

Одним из возможных методов задания графа $G(\mathbf{V},\mathbf{E})$ является перечень пар (k_i, k_j), произведение которых существует. Такой перечень удобно задавать в виде таблицы, в которой для каждой функции k_i как первого элемента пары указываются вторые элементы пары k_l^i, $l = 1,\ldots, q_i$. Набор отладочных тестов должен включать последовательное выполнение всех заданных в таблице пар функций.

4. Как отмечалось, выполнение одной и той же функции цифровой системы может осуществляться по-разному, с использованием различных ветвей программного обеспечения и различных режимов аппаратных средств. Поэтому выполнение каждой функции при выполнении набора отладочных тестов должно осуществляться с различными наборами признаков $\zeta_1, \zeta_2, \ldots, \zeta_n$. Выбранные различные сочетания признаков определяют количество необходимых выполнений функции k_i.

Заключение. При формировании множества тестовых примеров для отладки проектов цифровых микроэлектронных систем необходимо, чтобы система тестов содержала все возможные произведения функций k_i, k_j, а также кратное выполнение всех функций k с заданными наборами признаков $\zeta_1, \zeta_2, \ldots, \zeta_n$, т. е. выполнение каждой функции k столько раз, сколько наборов признаков для нее определено.

Разработка практического алгоритма формирования минимального набора отладочных тестов проектов цифровых микроэлектронных систем на основе проведенного теоретического исследования является текущей задачей.

Литература

1. ***Keresztes P., Tukacs A., Török M.*** A multi valued logic VHDL package for switch level simulation of novel digital CMOS Circuits // 2018 Inter. Conf. on Recent Innovations in Electrical, Electronics & Communication Engineering (Bhubaneswar, India, 2018). 2018. P. 25–28.

2. ***Lin Yi-Li, Su Alvin W.Y.*** Functional verification for SoC software/hardware Co-design: from virtual platform to physical platform // 2011 IEEE International SOC Conference. 2011. P. 201–206.

3. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design / ***Shi Jin, Liu, Weichao, Jiang Ming el al.*** // 2013 IEEE International Conference on Intelligent Rail Transportation. 2013. P. 71–74.

4. ***Nguen M.D.*** Hardware/software formal co-verification using hardware verification techniques // Fourth Int. Conf. on Communications and Electronics. 2012. P. 465–470.

5. Выбор платформ прототипирования для СФ-блоков и подсистем СНК / ***Е.М. Абрамов, А.В. Егоров, А.О. Козлов и др.*** // Вопросы радиоэлектроники. 2017. № 8. С. 76–83.

6. Методы и алгоритмы для логико-топологического проектирования микроэлектронных схем на вентильном и межвентильном уровне для перспективных технологий с вертикальным затвором транзистора / ***Г.А. Иванова, Д.И. Рыжова, С.В. Гаврилов и др.*** // Микроэлектроника. 2019. Т. 48. № 3. С. 201–210.

7. ***Иванников А.Д., Стемпковский А.Л.*** Формализация задачи отладки проектов цифровых систем // Информационные технологии. 2014. № 9. С. 3–10.

8. ***Оллонгрен А.*** Определение языков программирования интерпретирующими автоматами. М.: Мир, 1977. 288 с.

9. ***Мальцев А.И.*** Алгебраические системы. М.: Наука. 1970. 392 с.

10. ***Иванников А.Д.*** Формирование отладочного набора тестов для проверки функций цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2017. Т. 18. № 12. С. 795–801.

Поступила в редакцию 22.04.2020 г.; после доработки 09.06.2020 г.; принята к публикации 16.06.2020 г.

Иванников Александр Дмитриевич – доктор технических наук, профессор, главный научный сотрудник Института проблем проектирования в микроэлектронике Российской академии наук (Россия, 124365, г. Москва, г. Зеленоград, ул. Советская, 3), adi@ippm.ru

Стемпковский Александр Леонидович – доктор технических наук, профессор, академик Российской академии наук, научный руководитель Института проблем проектирования в микроэлектронике Российской академии наук (Россия, 124365, Москва, г. Зеленоград, ул. Советская, 3), ippm@ippm.ru

References

1. Keresztes P., Tukacs A., Török M. A Multi Valued Logic VHDL Package for Switch Level Simulation of Novel Digital CMOS Circuits. *2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE)*, Bhubaneswar, India, 2018, pp. 25–28.

2. Lin Yi-Li, Su Alvin W.Y. Functional Verification for SoC Software/Hardware Co-Design: From Virtual Platform to Physical Platform. *2011 IEEE International SOC Conference (SOCC)*, 2011, pp. 201–206.

3. Shi Jin, Liu Weichao, Jiang Ming el al. Software hardware co-simulation and co-verification in safety critical system design. *2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT). 2013,* pp. 71–74.

4. Nguen M.D. Hardware/software formal co-verification using hardware verification techniques. *Fourth Int. Conf. on Communications and Electronics (ICCE).* 2012, pp. 465–470.

5. Abramov E.M., Egorov A.V., Kozlov A.O., Poperechniy P.S., Putrya F.M., Frolova S.E. *Prototype Platform Choosing for IP Blocks and SoC Subsystems. Voprosi Radioelectroniki = Issues of radio electronic*s, 2017, no. 8, pp. 76–83. (in Russian).

6. Ivanova G.A., Rizhova D.I., Gavrilov S.V., Vasilev N.O., Stempkovskiy A.L. Methods and algorithms of microcircuits gate and intergate level logic-topological design for modern technologies with transistor vertical gate. *Microelectronika = Microelectronics*, 2019, vol. 48, no. 3, pp.201–210. (in Russian).

7. Ivannikov A.D., Stempkovsky A.L. Formal model of digital system design debugging task. *Informacionnye tehnologii = Information technologies***,** 2014, no. 9, pp. 3–10. (in Russian).

8. Ollongren A. *Programming languages definition by interpretive automata*. Moscow, Mir Publ., 1977. 288 p. (in Russian).

9. Maltcev A.I. *Algebraic systems*. Moscow, Nauka Publ., 1970. 392 p. (in Russian).

10. Ivannikov A.D. Debugging input set generation for testing of control digital systems functions. *Mekhatronika, Avtomatizatsiya, Upravlenie*, 2017, vol. 18, no. 12, pp. 795–801. (in Russian).

Received 22.04.2020; Revised 09.06.2020; Accepted 16.06.2020.

Information about the authors:

Alexander D. Ivannikov – Dr. Sci. (Eng.), Prof., Head Scientist of CAD Department of Institute for Design Problems in Microelectronics of Russian Academy of Sciences (Russia, 124365, Moscow, Zelenograd, Sovetskaya st., 3), adi@ippm.ru

Alexander L. Stempkovskiy – Dr. Sci. (Eng.), Prof., Member of Russian Academy of Sciences, Scientific Supervisor of Institute for Design Problems in Microelectronics of Russian Academy of Sciences (Russia, 124365, Moscow, Zelenograd, Sovetskaya st., 3), ippm@ippm.ru