УДК 681.3.02 DOI: 10.31114/2078-7707-2020-1-20-25

Выбор набора тестов функций цифровых систем для контроля правильности проектов

А.Д. Иванников

Институт проблем проектирования в микроэлектронике РАН, г. Москва

adi@ippm.ru

Аннотация — **При проектировании цифровых систем для проверки правильности проекта осуществляется составление некоторого набора тестовых примеров, которые подаются на компьютерную модель проекта цифровой системы. При этом осуществляется моделирование работы цифровой системы, задаваемое поданным входным воздействием, результаты моделирования проверяются разработчиком. В работе описана модель цифровых систем, используемая в этом процессе, на основе теории множеств, теории стационарных динамических систем и теории отношений эквивалентности анализируется пространство допустимых входных воздействий исходя из выполняемых цифровой системой функций. Анализируются цифровые системы, осуществляющие при своем функционировании выполнение последовательности функций из конечного алфавита. Предлагается алгоритм выбора тестовых примеров для проверки правильности (отладки) проектов цифровых систем при выполнении последними каждой из заданных функций. Алгоритм основан на выделении разработчиком классов эквивалентности в множестве входных воздействий, вызывающих выполнение цифровой системой конкретной функции из заданного алфавита.**

Ключевые слова — **автоматизация проектирования, цифровые системы, проверка правильности проекта, функционально-логическое моделирование, отладочные тесты.**

I. Введение

Еще в 90-е годы во многих работах указывалось на перспективность использования информационных технологий в различных областях современной человеческой деятельности: коммуникациях, образовании, науке и технике [1-3]. И действительно, в настоящее время проектирование современных цифровых микроэлектронных систем возможно только на основе систем автоматизированного проектирования. На компьютерную модель цифровой системы подаются некоторые входные воздействия, а реакция модели проектируемой системы проверяется на соответствие техническому заданию.

При этом важной задачей является выбор конечного числа конечных по времени тестовых входных взаимодействий (тестовых примеров). С ростом сложности проектируемых цифровых систем и, соответственно, ростом сложности и длительности тестирования их проектов все более актуальной становится задача выбора минимального полного в определенном смысле набора тестов, правильное выполнение которого позволяет убедиться в отсутствии ошибок проектирования [4, 5]. При решении этой задачи прежде всего необходимо выбрать уровень модели цифровой управляющей системы. Обычно осуществляется декомпозиция задачи отладки проекта [6], прежде всего по типу выявляемых ошибок. Так, для верификации временных диаграмм обмена информации между блоками используются модели цифровых элементов с многозначным представлением электрических сигналов [7, 8]. Для проверки правильности логического функционирования используются модели с булевым представлением сигналов [9, 10]. Используются также различные высокоуровневые модели.

Целью настоящей работы является формализация выбора тестовых примеров для отладки проектов цифровых систем исходя из перечня выполняемых ими функций, что не требует наличия детализированной формальной спецификации на функционирование проектируемой цифровой системы.

II. Используемая модель цифровых систем

Взаимодействие цифровой системы с объектом управления и внешним миром вообще осуществляется через внешние линии и шины – наборы линий, по которым передается однородная информация, например, адреса или данные. Причем в цифровых системах управления широко используются двунаправленные шины и линии, имеющие также состояние с высоким выходным сопротивлением (отключенное состояние). Будем рассматривать логическую модель сигналов на шинах и линиях цифровых систем, то есть считать, что значения сигналов представляются как 0 или 1 на линиях и как число из диапазона $0\text{-}2^n$ на шинах системы. Цифровые сигналы внешних шин и линий назовем терминальными переменными – множество $\mathbf{P}$. Переменная $p \in \mathbf{P}$ всегда имеет одно из значений конечного множества $\mathbf{Z}_p$, элементы которого определяют как целочисленное значение сигнала, так 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,

включая управляемый объект, есть последовательность переключений сигналов на терминальных шинах и линиях, то есть последовательность событий. Для каждой проектируемой системы имеется множество $\mathbf{\Psi}$ допустимых взаимодействий с внешней средой, каждое из которых есть отображение ψ: $[0,t)\rightarrow\mathbf{Q}$, $t\in\mathbf{T}$, $\mathbf{Q}=\prod_{p\in\mathbf{P}}\mathbf{Z}_p$.

В цифровых системах для каждого конечного временного интервала количество событий по терминальным переменным, то есть количество изменений их значений, конечно. В связи с этим любое взаимодействие ψ может быть представлено в виде вектора $(z^{\text{н}}_{p_1,\dots,}z^{\text{н}}_{p_k})$ начальных значений переменных $p_1,\dots, p_k$ (k – мощность множества **P**) в момент времени $t = 0$ и последовательности событий по переменным множества **P** с конечным числом событий за любой конечный интервал времени.

Если в последовательности событий выделить только события, являющиеся изменениями входных сигналов, то такую последовательность можно назвать входным воздействием. Однако, часто моменты подачи входных сигналов на цифровую систему определяются готовностью системы принять эти сигналы, на что указывают определенные выходные сигналы системы. Выполнение какой-либо операции, например, считывания данных цифровой системой, может инициироваться не сигналами внешней среды, а самой системой. В связи с этим использование в качестве аргументов функционирования цифровой системы входных воздействий не всегда удобно.

Выделим из последовательности событий взаимодействия ψ последовательность входных событий и выходных событий управления обменом, которые по заданному протоколу обмена обуславливают моменты времени входных событий. Назовем эту последовательность входным взаимодействием:

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

где $p_{i_1}, p_{i_2}, p_{i_3},\dots$ - переменные, принадлежащие множеству **P;**

z_{j_1}, z_{j_3}, z_{j_5},… - значения переменных непосредственно перед событием;

z_{j_2}, z_{j_4}, z_{j_6},… - значения переменных непосредственно после события;

$\chi^{t_1}_{p_{i_1},z_{j_1},z_{j_2}}$, $\chi^{t_2}_{p_{i_2},z_{j_3},z_{j_4}}$, $\chi^{t_3}_{p_{i_3},z_{j_5},z_{j_6}}$ … - входные события и выходные события управления обменом;

$t_1\leq t_2\leq t_3\leq$ … - упорядоченная последовательность времен событий входного взаимодействия;

n – количество входных переменных;

q – количество переменных управления обменом.

В рассматриваемой модели в качестве аргументов функционирования цифровых систем используются входные взаимодействия, что дает возможность рассматривать режимы работы, инициируемые как внешними входными сигналами, так и самими цифровыми системами. Более подробно формальное представление допустимых взаимодействий цифровых систем рассмотрено в [11].

III. Структура множества допустимых входных взаимодействий

Пусть для некоторой проектируемой цифровой системы заданы входные взаимодействия

$$\mu_1\colon[0,t_1)\rightarrow\mathbf{U}\times\mathbf{Y}^{\text{o}},\ \ \mu_2\colon[0,t_2)\rightarrow\mathbf{U}\times\mathbf{Y}^{\text{o}},\quad t_1\in\mathbf{T}_1, t_2\in\mathbf{T}_2,$$

где **U** – множество состояния выходных переменных;

$\mathbf{Y}^{\text{o}}$- множество состояния выходных переменных управления обменом.

Определим произведение $\mu_{1,2}=\mu_1\cdot\mu_2$, $\mu_{1,2}\colon[0,t_1+t_2)\rightarrow\mathbf{U}\times\mathbf{Y}^{\text{o}}$ как

$$\mu_{1,2}=\mu_1\cdot\mu_2=\begin{cases}\mu_1(t)\text{ при }t\in[0,t_1)\\ \mu_2(t-t_1)\text{ при }t\in[t_1,t_1+t_2)\end{cases},$$

также являющееся входным взаимодействием.

Тогда на множестве входных взаимодействий **M** определена в общем случае частичная мультипликативная полугруппа $<\mathbf{M},\cdot>$.

Практика показывает, что каждая цифровая система выполняет некоторую последовательность функций из конечного алфавита **K**, причем выполнение каждой функции вызывается одним из входных взаимодействий определенного класса. Из этого утверждения следует, что полугруппа $<\mathbf{M},\cdot>$ имеет бесконечное множество $\overline{\mathbf{M}}$ порождающих элементов, причем $\overline{\mathbf{M}}=\bigcup_{k\in\mathbf{K}}\mathbf{M}^k$, где $\mathbf{M}^k$ – множество входных взаимодействий, обуславливающих выполнение цифровой системой функции k.

С точки зрения влияния на поведение цифровой системы различия двух входных взаимодействий μ_1 и μ_2, $\mu_1\in\overline{\mathbf{M}}$, $\mu_2\in\overline{\mathbf{M}}$ могут быть в той или иной степени «невелики». Так, одно из входных взаимодействий может содержать, а другое не содержать одно или несколько безразличных событий, никак не влияющих на функционирование цифровой системы. Примером несущественного события может служить следующая ситуация. После того, как осуществлено считывание информации с какого-либо входа и до момента, когда цифровая система может опять обратиться к этому же источнику информации, сигнал на этом входе может переключиться в любое состояние или содержать любое количество произвольных переключений. При другом критерии «близости» «почти одинаковыми» являются входные взаимодействия, содержащие равное количество событий, отличающихся только значениями ряда t_j в пределах допустимых ограничений. Могут быть рассмотрены и более крупные группы входных взаимодействий.

Представим множество значений данных $\boldsymbol{\mathcal{D}}$ входных взаимодействий, обуславливающих выполнение цифровой системой определенной функции k, как $\boldsymbol{\mathcal{D}}=\bigcup_{i=1}^{n}\boldsymbol{\mathcal{D}}_i$, $\boldsymbol{\mathcal{D}}_i\cap\boldsymbol{\mathcal{D}}_j=\emptyset$ при $i\neq j$. В одну группу входных взаимодействий могут быть отнесены взаимодействия, данные которых принадлежат одному и тому же подмножеству $\boldsymbol{\mathcal{D}}_i$. Каждое подмножество $\boldsymbol{\mathcal{D}}_i$ в свою очередь может быть разбито на непересекающиеся

подмножества $\boldsymbol{\mathcal{D}}_{ij}$ и т.д. Разбиение множества значений данных должно осуществляться исходя из физического смысла задачи таким образом, чтобы входные взаимодействия с данными различных подмножеств $\boldsymbol{\mathcal{D}}_i$ обуславливали несколько различные алгоритмы их обработки. Самой большой группой являются множества $\mathbf{M}^k, k \in \mathbf{K}$ входных взаимодействий, обуславливающих выполнение цифровой системой функции k.

Математическим аналогом понятия «близости» входных взаимодействий является отношение эквивалентности. Пусть имеется конечное множество $\boldsymbol{\Lambda}$ отношений эквивалентности λ. В частности, множество $\boldsymbol{\Lambda}$ обязательно включает отношения эквивалентности λ', λ''.

$(\mu_1, \mu_2) \in \lambda'$ 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, обуславливающие выполнение функции k, в множество $\mathbf{M}^k$.

$(\mu_1, \mu_2) \in \lambda''$, 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, присутствующий в μ_1, μ_2, принадлежит одной и той же подобласти $\boldsymbol{\mathcal{D}}_i$ данных, где $\bigcup_{i=1}^{n} \boldsymbol{\mathcal{D}}_i = \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{D}}_i \cap \boldsymbol{\mathcal{D}}_j = \emptyset$ при $i \neq j$, $\boldsymbol{\mathcal{D}}$ - область данных для $\mathbf{M}^k$. Так возможен случай, когда $(\mu_1, \mu_2) \in \lambda''$ 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, обрабатываемым по одинаковому алгоритму, и изменяющим 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 $\boldsymbol{\mathcal{D}}$.

В случае, когда функция k предусматривает периодический фрагмент последовательности событий, который может повторяться в различных входных взаимодействиях множества $\mathbf{M}^k$ различное число раз (например, ввод различного числа слов информации), может быть использовано отношение эквивалентности λ'''. $(\mu_1, \mu_2) \in \lambda'''$, если μ_1 и μ_2 содержат одинаковое количество периодически повторяющихся фрагментов. Могут быть заданы и другие отношения эквивалентности.

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

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

Если на некотором множестве $\mathbf{M}^*$ (в качестве $\mathbf{M}^*$ может выступать $\overline{\mathbf{M}}$, $\mathbf{M}^k$ или класс эквивалентности $\mathbf{M}^k$ по отношению эквивалентности λ) задано отношение λ_1, $\lambda_1 \in \boldsymbol{\Lambda}$, то существует разбиение $\mathbf{M}^*$ на классы эквивалентности $\mathbf{M}^{*i}$ по отношению λ_1. Если на некотором $\mathbf{M}^{*i}$ определено отношение λ_2, $\lambda_2 \in \boldsymbol{\Lambda}$, то для μ, где $\mu \in \mathbf{M}^{*i}$, существует произведение эквивалентностей $\lambda_1 \cdot \lambda_2$, которое всегда является эквивалентностью [3]. Произведения эквивалентностей $\prod_{\lambda_i \in \boldsymbol{\Lambda}'} \lambda_i$, где $\boldsymbol{\Lambda}' \subseteq \boldsymbol{\Lambda}$, которые всегда являются эквивалентностями, позволяют провести классификацию множества $\overline{\mathbf{M}}$ с той степенью подробности, которая необходима разработчику, определившему множество $\boldsymbol{\Lambda}$. Входные взаимодействия каждого класса эквивалентности, определяемого максимально возможными произведениями $\prod_{\lambda_i \in \boldsymbol{\Lambda}'} \lambda_i$, где $\boldsymbol{\Lambda}' \subseteq \boldsymbol{\Lambda}$, различаются только значениями ряда t_j в пределах, не нарушающих ограничений на допустимые времена событий.

Как указывалось выше, самыми крупными классами эквивалентности на множестве $\overline{\mathbf{M}}$ являются множества $\mathbf{M}^k$, соответствующие выделению алфавита выполняемых функций $\mathbf{K}$.

IV. Составление набора отладочных тестов

Набор отладочных тестов должен проверять как возможность выполнения одной функции после другой, если такая последовательность является допустимой, так и правильность выполнения самих функций. При этом правильность выполнения самих функций является основой. Совершенно ясно, что большее количество отладочных тестов позволяет говорить о правильной работе отлаживаемой системы с большей уверенностью. В связи с тем, что большее количество отлаживаемых тестов повышает трудоемкость и временные затраты на отладку, необходимо, чтобы при увеличении количества тестовых примеров все они были наиболее информативными.

Как указывалось выше, для входных взаимодействий, вызывающих выполнение одной функции, определено множество $\boldsymbol{\Lambda}$ отношений эквивалентности. Каждому λ_i, где $\lambda_i \in \boldsymbol{\Lambda}$, поставим в соответствие признак $\mathcal{L}_i$ - переменную с конечным множеством значений $\mathbf{Z}^*_i$. Каждое значение z^*_i, $z^*_i \in \mathbf{Z}^*_i$ признака $\mathcal{L}_i$ указывает на принадлежность μ, $\mu \in \mathbf{M}$, j-ому классу эквивалентности входных взаимодействий по λ_i. Разработчик задает не множество эквивалентностей $\boldsymbol{\Lambda}$, а набор признаков $\mathcal{L}_1$, $\mathcal{L}_2$, …, $\mathcal{L}_n$ и алфавиты их значений. Множество значений признака $\mathcal{L}_1$ всегда есть $\mathbf{K}$. Признак $\mathcal{L}_1$ определяет разбиение $\overline{\mathbf{M}}$ на $\mathbf{M}^k$.

Сформулируем задачу составления набора отладочных тестов для проверки отдельных функций формально. Пусть имеется конечное множество признаков

$\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$, каждый из которых может принимать конечное множество значений $\mathbf{Z}^*_1, \mathbf{Z}^*_2, \ldots, \mathbf{Z}^*_n$. Пусть признаки $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$ упорядочены таким образом, что для каждого $\mathcal{L}_l$, $l=2,\ldots,n$ существует один и только один признак $\mathcal{L}_i$, $i< l$ такой, что значение $\mathcal{L}_l$ для входного взаимодействия μ определено в том и только в том случае, если признак $\mathcal{L}_i$ для μ имеет значение $z_i^{j_l}, z_i^{j_l} \in \mathbf{Z}_i^*$. Тогда взаимосвязь признаков можно представить двудольным графом $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$. Множество вершин этого графа есть $\boldsymbol{\mathcal{V}} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\} \cup \{\cup_{i=1}^n \mathbf{Z}_i^*\}$. Одна группа вершин представляет множество признаков, а другая группа – возможные значения каждого признака. Множество ребер $\boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{E}}^1 \cup \boldsymbol{\mathcal{E}}^2$, $\boldsymbol{\mathcal{E}}^1 \cap \boldsymbol{\mathcal{E}}^2 = \emptyset$. Ребра множества $\boldsymbol{\mathcal{E}}^1$ соединяют вершину $\mathcal{L}_i$ с вершиной z, если $z \in \mathbf{Z}_i^*$. Ребра множества $\boldsymbol{\mathcal{E}}^2$ соединяют вершину z_i^j, где $z_i^j \in \mathrm{Z}_i^*$, с вершиной $\mathcal{L}_l$, если признак $\mathcal{L}_l$ определен для входного взаимодействия μ в том случае, если для этого входного взаимодействия $\mathcal{L}_i = z_i^j$.

Пусть множество отладочных тестов считается полным, если для любого признака $\mathcal{L}_i, i = 1, \ldots, n$ и $z \in \mathbf{Z}_i^*$ в множестве отладочных тестов найдется по крайней мере одно входное взаимодействие, для которого $\mathcal{L}_i = z$. Минимальным полным множеством отладочных тестов назовем такое полное множество тестов, число тестов в котором минимально. Задача составления минимального полного множества отладочных тестов состоит в выборе сочетаний значений признаков для каждого отладочного теста множества.

V. Алгоритм формирования минимального множества отладочных тестов

1. Присвоим ранги вершинам $\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n$, равные количеству ребер из множества $\boldsymbol{\mathcal{E}}^2$, входящих в путь от $\mathcal{L}_1$ до $\mathcal{L}_i$. Вершина $\mathcal{L}_1$ имеет ранг 0. Поскольку в $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ каждая вершина достижима из вершины $\mathcal{L}_1$, то все вершины $\mathcal{L}_1, \ldots, \mathcal{L}_n$ имеют ранг, причем ранг 0 имеет только одна вершина $\mathcal{L}_1$.

2. Среди вершин множества $\{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, имеющих максимальный ранг, выделим вершину $\mathcal{L}_{i_1}$ такую, что $|\mathbf{Z}_{i_1}^*|$ максимально. Рассмотрим подграф $\mathcal{G}_{i_1}$ графа $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$, который включает вершину $\mathcal{L}_{i_1}$, вершины $z_{i_1}, z_{i_1} \in \mathbf{Z}_{i_1}^*$, вершину z_l, $z_l \in \mathbf{Z}_l^*$, связанную с вершиной $\mathcal{L}_{i_1}$ ребром из множества $\boldsymbol{\mathcal{E}}^2$, вершины $\mathcal{L}_{i_2}, \ldots, \mathcal{L}_{i_m}$, связанные с вершиной z_l ребрами из множества $\boldsymbol{\mathcal{E}}^2$ (если такие вершины имеются), вершины z, где $z \in \mathbf{Z}_{i_2}^*$ или $z \in \mathbf{Z}_{i_3}^*$ … или $z \in \mathbf{Z}_{i_n}^*$, а также ребра, соединяющие перечисленные вершины. Все вошедшие в подграф $\mathcal{G}_{i_1}$ вершины из множества $\{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$ имеют максимальный ранг.

Подграфу $\mathcal{G}_{i_1}$ поставим в соответствие $|\mathbf{Z}_{i_1}^*|$ наборов значений признаков $(z_{i_1}, z_{i_2}, \ldots, z_{i_m})$, $z_{i_1} \in \mathbf{Z}_{i_1}^*$, $z_{i_2} \in \mathbf{Z}_{i_2}^*, \ldots, z_{i_m} \in \mathbf{Z}_{i_m}^*$. Признак $\mathcal{L}_{i_1}$ в первом наборе имеет значение $z_{i_1}^1$, во втором наборе $z_{i_1}^2$, и так далее. В последнем наборе признак $\mathcal{L}_{i_1}$ имеет значение $z_{i_1}^k$, где $k = |\mathbf{Z}_{i_1}^*|$. Признаки $\mathcal{L}_{i_s}, s = 2, \ldots, m$ в первом наборе имеют значения $z_{i_s}^1$, и в каждом последующем $z_{i_s}^{(p+1)_{mod|\mathbf{Z}_{i_s}^*|}}$, где p – номер значения признака $\mathcal{L}_{i_1}$ в предыдущем наборе.

В случае, если подграф $\mathcal{G}_{i_1}$ содержит только одну вершину из множества $\{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, а именно $\mathcal{L}_{i_s}$, наборы значений признаков есть $(z_{i_1}^1), (z_{i_1}^2), \ldots, (z_{i_1}^k)$.

Удалим из графа $\mathcal{G}(\mathcal{V}, \mathcal{E})$ подграф $\mathcal{G}_{i_1}$ и ребро из множества $\boldsymbol{\mathcal{E}}^1$, связывающее вершины z_l и $\mathcal{L}_l$. Добавим в полученный граф $|\mathbf{Z}_{i_1}^*|$ вершин, помеченных наборами значений признаков $(z_l, z_{i_1}, z_{i_2}, \ldots, z_{i_m})$, $z_l \in \mathbf{Z}_l^*$, причем для всех $|\mathbf{Z}_{i_1}^*|$ вершин значение z_l одинаково, а поднаборы $(z_{i_1}, z_{i_2}, \ldots, z_{i_m})$ равны наборам значений признаков, соответствующих подграфу $\mathcal{G}_{i_1}$. Соединим добавленные вершины с вершиной $\mathcal{L}_l$, полученные ребра отнесем к множеству $\boldsymbol{\mathcal{E}}^1$. Полученный граф примем за $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$.

3. Если $\mathcal{L}_l$ не есть $\mathcal{L}_1$, то повторим пункт 2. Если $\mathcal{L}_l$ есть $\mathcal{L}_1$, то полученное множество наборов значений признаков представляет отладочные тесты минимального полного множества.

VI. Анализ предложенного алгоритма

Покажем, что приведенный выше алгоритм всегда сходится, а полученное множество наборов значений признаков есть минимальное полное множество отладочных тестов.

Каждое выполнение шага 2 алгоритма уменьшает количество вершин множества $\{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$ в графе $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$. Следовательно, после конечного числа выполнения шага 2 граф $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ будет содержать только одну вершину из множества $\{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$, а именно вершину $\mathcal{L}_1$, что является признаком окончания работы.

Для каждого подграфа $\mathcal{G}_{i_1}$, выделяемого на шаге 2, наборы значений признаков будут содержать все возможные значения признаков $\mathcal{L}_{i_1}, \ldots, \mathcal{L}_{i_m}$, вошедших в подграф $\mathcal{G}_i$. Все наборы значений признаков подграфа $\mathcal{G}_{i_1}$ сохраняются в новом графе $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$. Следовательно, для всех признаков полученные наборы будут содержать все значения, то есть полученное множество наборов значений признаков будет полным множеством отладочных тестов.

Для каждого подграфа $\mathcal{G}_i$, выделяемого на шаге 2, полученное множество наборов признаков минимально и содержит $max|\mathbf{Z}_{i_k}^*|, k = 1, \ldots, m$ наборов. Эти наборы признаков могут войти в множество окончательных наборов только в сочетании со значением z_l признака $\mathcal{L}_l$, то есть значение z_l признака должно встречаться в полном множестве отладочных тестов не менее $max|\mathbf{Z}_{i_k}^*|, k = 1, \ldots, m$, раз. Следовательно, полученное в результате работы алгоритма множество отладочных тестов будет минимальным.

Минимальная мощность минимального полного множества отладочных тестов есть $max\,|\mathbf{Z}_i^*|, i =$

$1, \dots, n$, где n – количество признаков $\mathcal{L}_1, \dots, \mathcal{L}_n$; максимальная мощность минимального полного множества отладочных тестов есть $\sum_{i=1}^{n}(|\mathbf{Z}_i^*|-1)+1$. Покажем это.

Минимальное полное множество отладочных тестов не может содержать менее, чем $max\,|\mathbf{Z}_i^*|, i = 1, \dots, n$, тестов, так как в связи с нашим предположением для каждого признака множества $\{\mathcal{L}_1, \dots, \mathcal{L}_n\}$, в том числе для $\mathcal{L}_i$, для которого $|\mathbf{Z}_i^*|$ максимально, полное множество тестов должно содержать все возможные значения из $\mathbf{Z}_i^*$.

Минимальное полное множество будет содержать *max* $|\mathbf{Z}_i^*|$, *i*=1,…, *n*, отладочных тестов в том случае, когда в графе $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ все вершины $\mathcal{L}_2, \dots, \mathcal{L}_n$ будут непосредственно связаны с $\mathcal{L}_1$ ребрами подмножества $\mathcal{E}^2$.

При наличии только одного признака $\mathcal{L}_1$ минимальное полное множество отладочных тестов содержит $|\mathbf{Z}_1^*|$ тестов, при двух признаках $\mathcal{L}_1, \mathcal{L}_2$ - $|\mathbf{Z}_1^*| + |\mathbf{Z}_2^*| - 1$ тестов, при трех признаках $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ - не более $|\mathbf{Z}_1^*| + |\mathbf{Z}_2^*| - 1 + |\mathbf{Z}_3^*| - 1$ тестов и т.д. Таким образом, максимальная мощность минимального полного множества отладочных тестов равна $\sum_{i=1}^{n}(|\mathbf{Z}_1^*|-1)+1$. Минимальное полное множество отладочных тестов содержит $\sum_{i=1}^{n}(|\mathbf{Z}_1^*|-1)+1$ тестов в том случае, если в графе $\mathcal{G}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}})$ каждой вершине *z*, где $z \in \mathbf{Z}_i^*, i = 1, \dots, n$, инцидентно не более одного ребра из множества $\mathcal{E}^2$.

VII. Заключение

Предлагаемый алгоритм обеспечивает формирование минимального множества отладочных тестов на основе заданного разработчиком перечня отношений эквивалентности на множестве входных взаимодействий. Другими словами, разработчик выбирает разделение каждой функции, выполняемой цифровой системой, на подфункции, исходя из физического смысла. Безусловно, при этом присутствует некоторый субъективный фактор. При иерархическом разделении функций на подфункции разработчик в какой-то мере ориентируется на свое понимание о том, как данная подфункция реализуется в цифровой системе. Тем ни менее, выбор тестовых примеров для отладки проектов цифровых систем на основе проверки выполняемых функций является весьма эффективным.

Литература

[1] Юсупов Р.М. Информатизация и наука // Проблемы информатизации. 1994. № 1-2. С. 22-28.

[2] Тихонов А., Лобанов В., Иванников А. Время информатизации // Высшее образование в России. 1996. № 2. С. 30.

[3] Иванников А., Кривошеев А., Куракин Д. Развитие сети телекоммуникаций в системе высшего образования Российской Федерации // Высшее образование в России. 1995. № 2. С. 87.

[4] Слинкин Д.И. Анализ современных методов тестирования и верификации проектов сверхбольших интегральных схем // Программные продукты и системы. 2017. Т. 30. № 3. С. 401-408.

[5] Абрамов Е.М., Егоров А.В., Козлов А.О., Поперечный П.С., Путря Ф.М., Фролова С.Е. Выбор платформ прототипирования для СФ-блоков и подсистем СНК // Вопросы радиоэлектроники. 2017. № 8. С. 76-83.

[6] Иванников А.Д. Анализ методов декомпозиции задачи отладки проектов цифровых систем // Информационные технологии. 2016. Т. 22. № 10. С. 758-763.

[7] Гаврилов С.В., Гудкова О.Н., Щелоков А.Н. Логико-временной анализ нанометровых схем на основе интервального подхода // Известия ЮФУ. Технические науки. 2012. № 7(132). С. 85-91.

[8] Гаврилов С.В., Иванова Г.А. Анализ быстродействия сложных цифровых схем с учетом неопределенности технологических и схемных параметров // Вестник Рязанского государственного радиотехнического университета. 2015. № 53. С. 29-35.

[9] Кащеев Н.И., Пономарев Д.М., Подъяблонский Ф.М. Построение тестов цифровых схем с использованием обобщенной модели неисправностей и непрерывного подхода к моделированию // Вестник Нижегородского университета им. Н.И.Лобачевского. 2011. №3 (2). С. 72-77.

[10] Зотов В. Инструментальные средства разработки и отладки цифровых устройств и встраиваемых микропроцессорных систем, проектируемых на основе ПЛИС FPGA фирмы XILINX серии Kintex-7 // Компоненты и технологии. 2012. № 4 (129). С. 124-132

[11] Иванников А.Д., Стемпковский А.Л. Формализация задачи отладки проектов цифровых систем // Информационные технологии. 2014. № 9. С. 3-10.

Function Test Set Generation for Design Correctness Checking

A.D. Ivannikov

Institute for design problems in microelectronics of RAS, Moscow, adi@ippm.ru

***Abstract* — When designing digital systems to verify the correctness of a design, a certain set of test examples is compiled, which are fed to the computer model of the digital system design. The simulation results are checked by the developer. The paper describes the model of digital systems used in this process, on the basis of set theory, the theory of stationary dynamical systems and the theory of equivalence relations, analyzes the space of permissible input influences based on the functions performed by the digital system. Digital systems are analyzed that, during their functioning, perform a sequence of functions from a finite alphabet. An algorithm is proposed for selecting test cases for verifying the correctness (debugging) of digital systems designs when the latter perform each of the specified functions. The algorithm is based on the developer highlighting equivalence classes in the set of input actions that cause the digital system to perform a specific function from a given alphabet.**

***Keywords* — design automation, digital systems, design validation, functional-logical modeling, debugging tests**

REFERENCES

[1] Usupov R.M. Informatizacia I nauka (Informatization and Science) // Problemi informatizacii. 1994. № 1-2. P. 22-28.

[2] Tikhonov A., Lobanov V., Ivannikov A. Vremja informatizacii (Time of Informatization) // Vishee obrazovanie v Rossii. 1996. № 2. P.30.

[3] Ivannikov A., Krivosheev A., Kurakin D. Razvitie seti teleommunikaciy v systeme vishego obazovaniya Rossiyskoy Federacii (Telecommunication net development in higher education system of Russian Federation) // Vishee obrazovanie v Rossii. 1995. № 2. P.87.

[4] Slinkin D.I. Analys sovremennih metodov testirovaniya I verifikacii proektov sverhbolshih integralnih shem (Analisys of modern test and verification methods for VLSI design) \\ Programmnie producti I systemi. 2017. V. 30. №3. P. 401-408.

[5] Abramov E.M., Egorov A.V., Kozlov A.O., Poperechniy P.S., Putrya F.M., Frolova S.E. Vibor platform prototipirovaniya dlya SF-blokov i podsystem SnK (Prototype Platform Choosing for IP Blocks and SoC Subsystems) // Voprosi Radioelectroniki. 2017. № 8. P. 76-83.

[6] Ivannikov A.D. Analiz metodov dekompozicii zadachi otladki proektov cifrovih system (Decomposition Methods Analisys for Digital System Design Debugging). Informacionnie Technologii. 2016. Vol. 22. No. 10. P. 758-763.

[7] Gavilov S.V., Gudkova O.N., Shelokov A.N. Logiko-vremennoy analiz nanometrovih shem na osnove itervalnogo podhoda (Nanometric ircuit logic timing analisys based on interval approach) // Izvestiya UFU. Tehnicheskie nauki. 2012. № 7(132). P. 85-91.

[8] Gavrilov S.V., Ivanova G.A. Analis bistrodeistviya slojnih cifrovih chem s uchetom neopredelennisti tehnologicheskih I shemnih parametrov (Timinп analisys of complex digital circuits with technological and circuit parameters uncertainty) // Vestnik Rusanskogo gosudarstvennogo radiotehnicheskogo universiteta. 2015. № 53. P. 29-35.

[9] Kasheev N.I., Ponomarev D.M., Podyablonsky F.M. Postroenie testov cifrovih chem. C ispolzovaniem obobshennoi modeli neispravnostei i neprerivnogo podhoda k modelirovaniu (Digital Circuits Test Generation Based on Generalized Malfunction Model and Continuous Simulation Approach). Vestnik Nijegorodskogo Universiteta. 2011. № 3(2). P. 72-77.

[10] Zotov V. Instrumentalnie sredstva razrabotki i otladki cifrovih ustroistv i vsraivaemih mikroprocessornih system, proektiruemih na osnove PLIS FPGA firmi XILINX serii Kintex-7 (Design and Debugging Means for Digital Devices and In-Circuit Microprocessor Systems on the Bases of XILINX FPGA Kintex-7 Series) // Komponenti i Technologii. 2012. № 4 (129). P. 124-132.

[11] Ivannikov A.D., Stempkovsky A.L. Formalizaciya zadachi otladki proektov cifrovih system (Formal Model of Digital System Design Debugging Task) // Informacionnie Technologii. 2014. № 9. P. 3-10.