Верификация VHDL-описаний цифровых устройств, представленных в виде композиции управляющего и операционного блоков

Часть 1. Верификация на основе покрытия VHDL-кода

Николай Авдеев, Пётр Бибило (bibilo@newman.bas-net.by)

В статье описана методика проведения верификации VHDL-описаний цифровых устройств, представленных в виде композиции управляющего и операционного блоков. Под верификацией понимается проверка соответствия VHDL-описания проектируемого цифрового устройства спецификациям на проектирование. Для верификации используется логическое моделирование в системе Questa Sim (Mentor Graphics), выполняемое с покрытием VHDL-кода.

Введение

Цифровые устройства, представляемые в виде композиции (соединения) управляющего и операционного блоков, давно нашли широкое применение в практике проектирования [1]. В настоящее время проекты таких устройств задаются на языках VHDL и Verilog [2], предназначенных для проектирования цифровых схем на современной базе заказных СБИС (сверхбольших интегральных схем) либо программируемых пользователями логических интегральных схем типа FPGA.

По VHDL-описаниям проектов автоматически строятся синхронные логические схемы в том или ином базисе логических элементов, называемом технологическим (целевым) базисом либо целевой библиотекой логических элементов. В настоящее время процесс синтеза автоматизирован и важнейшей проблемой при создании проектов СБИС и систем на кристалле [3] является проблема верификации исходных спецификаций, представленных на VHDL либо других языках, используемых для алгоритмического описания проектируемых цифровых устройств и систем.

Под верификацией понимается проверка правильности исходного VHDL-описания, т.е. проверка соответствия составленного синтезируемого VHDL-описания проектируемой цифровой системы спецификациям на проектирование. Далее будут использоваться примеры VHDL-описаний устройств, однако применяемая методика и инструментальные средства применимы и к описаниям на языке Verilog.

Цифровые устройства рассматриваемого типа, как правило, входят в состав более сложных проектов. Для проведения верификации всего проекта в целом требуется провести сначала отдельную верификацию таких устройств. Проведение верификации VHDL-описаний цифровых устройств, представленных в виде композиции управляющего и операционного блоков, является целью данной статьи. В первой части статьи описывается маршрут моделирования в системе Questa Sim [4], позволяющий выполнить верификацию на основе моделирования проекта с покрытием VHDL-кода, вторая часть будет посвящена верификации на основе функционального покрытия.

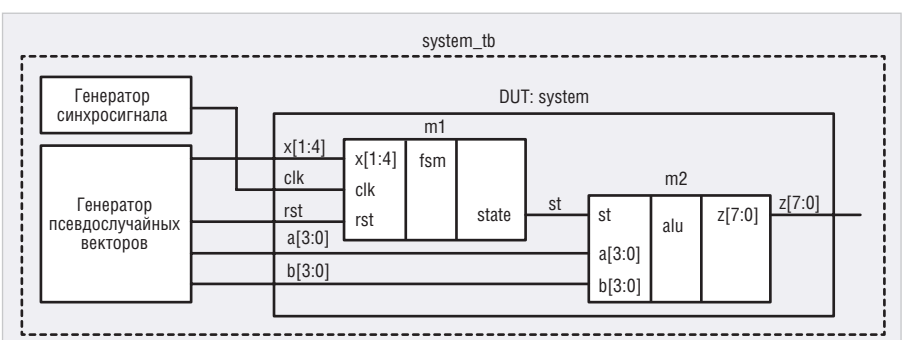

Рис. 1. Цифровое устройство *system*

Пример операционного устройства

Простой пример цифрового устройства system представлен на рисунке 1. Описание устройства на языке VHDL представлено в листинге 1. Устройство состоит из двух подсхем (блоков): управляющий блок является конечным автоматом и имеет имя *fsm* (листинг 2), операционный блок – имя *alu* (листинг 3).

Управляющий автомат *fsm* задан графом *G* переходов (см. рис. 2), функ-

Листинг 1

```
package system_pkg is
type st_t is (s1, s2, s3, s4, s5, s6);
end system_pkg;
library ieee;
use ieee.std_logic_1164.all;
use work.system_pkg.all;
entity system is
port(x   : in  std_logic_vector(1 to 4);
clk, rst : in  std_logic;
a, b: in  std_logic_vector(3 downto 0);
z   : out std_logic_vector(7 downto 0));
end;
architecture str of system is
component fsm
port (
x     : in  std_logic_vector(1 to 4);
clk, rst : in  std_logic;
state : out st_t);
end component;
component alu
port (
a, b : in  std_logic_vector(3 downto 0);
st   : in  st_t;
z    : out std_logic_vector(7 downto
0));
end component;
signal st : st_t;
begin
m1 : fsm port map (x, clk, rst, st);
m2 : alu port map (a, b, st, z);
end;
```

Листинг 2

```
library ieee;
use ieee.std_logic_1164.all;
use work.system_pkg.all;
entity fsm is
port(x : in  std_logic_vector(1 to 4);
clk, rst : in  std_logic;
state    : out st_t);
end fsm;
architecture beh of fsm is
signal n_st, st : st_t;
begin
n_st_p : process (st, x)
begin
case st is
when s1 =>
n_st <= s2;
when s2 =>
if ((x(1) and not x(2) and not x(3))
or (x(1) and x(2))) = '1' then
n_st <= s3;
elsif ((x(1) and
not x(2) and x(3)) = '1') then
n_st <= s4;
elsif (not x(1) = '1') then
n_st <= s5;
end if;
when s3 => n_st <= s4;
when s4 =>
if (not x(2) = '1') then
n_st <= s1;
elsif (x(2) = '1') then
n_st <= s6;
end if;
when s5 =>
if ((not x(1) and x(4)) = '1') then
n_st <= s1;
elsif ((not x(4)) or
(x(1) and x(4))) = '1' then
n_st <= s4;
end if;
when s6 =>
n_st <= s1;
end case;
end process n_st_p;
st_p : process (clk, rst)
begin
if rst = '1' then
st <= s1;
elsif clk'event and clk = '1' then
st <= n_st;
end if;
end process st_p;
state <= st;
end beh;
```

Листинг 3

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use work.system_pkg.all;
entity alu is
port (
a, b : in  std_logic_vector(3 downto 0);
st   : in  st_t;
z    : out std_logic_vector(7 downto
0));
end alu;
architecture beh of alu is
signal z_u : unsigned(7 downto 0);
signal a_u, b_u : unsigned(3 downto 0);
begin
a_u <= unsigned(a);
b_u <= unsigned(b);
z_u <=
RESIZE(a_u and b_u, 8)  when st = s1
else
RESIZE(a_u or  b_u, 8)  when st = s2
else
RESIZE(a_u xor b_u, 8)  when st = s3
else
RESIZE(a_u xnor b_u, 8) when st = s4
else
RESIZE(a_u+('0'&b_u),8) when st = s5
else
RESIZE(a_u * b_u, 8)    when st = s6
else
(others => 'X');
z  <= std_logic_vector(z_u);
end beh;
```

ции переходов даны в таблице 1. Если переход является безусловным, то ДНФ, задающая соответствующее условие перехода, равна единице.

На рисунке 2 не показаны асинхронные переходы из любого состояния s_i в начальное состояние s_1 при единичном значении сигнала сброса *rst*. Двоичные входные векторы (порты) *a, b* называются операндами операционного блока. В примере число разрядов каждого из операндов равно четырём. Функции операционного блока заданы в таблице 2.

В зависимости от состояния функция операционного блока может быть либо арифметической (сложение либо умножение операндов *a, b,* понимаемых как двоичные коды чисел без знака), либо логической – в этом случае логическая операция выполняется над соответствующими разрядами двоичных векторов *a, b*.

Функционирование VHDL-модели управляющего автомата *fsm* осуществляется по тактам, смена состояния выполняется по переднему фронту синхросигнала *clk*. Управляющий автомат *fsm* начинает функционирование из начального состояния s_1, меняет свои состояния и всегда возвращается в начальное состояние. Заметим, что для других цифровых устройств такого рода в графе переходов могут быть петли, что означает, что управляющий автомат не выходит из текущего состояния, а ожидает требуемую комбинацию управляющих входных сигналов x_1, x_2, x_3, x_4, чтобы перейти в другое состояние. В процессе функционирования цепочки состояний управляющего автомата образуют на графе *G* различные циклы. В данном простом примере каждый переход автомата из одного состояния в другое вызывает смену выполнения соответствующей операции в операционном блоке.

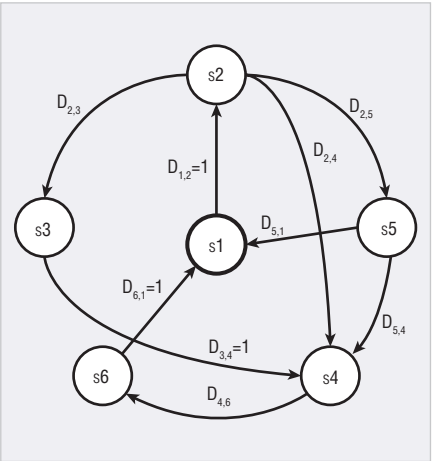

Рис. 2. Граф *G* переходов управляющего автомата *fsm*

Таблица 1. Переходы управляющего автомата *fsm*

s_i	s_j	Условие перехода
s_1	s_2	$D_{1,2}=1$
s_2	s_3 s_4	$D_{2,3}=x_1\bar{x}_2\bar{x}_3 \vee x_1x_2$ $D_{2,4}=x_1\bar{x}_2x_3$
	s_5	$D_{2,5}=\bar{x}_1$
s_3	s_4	$D_{3,4}=1$
s_4	s_1 s_6	$D_{4,1}=\bar{x}_2$ $D_{4,6}=x_2$
s_5	s_1 s_4	$D_{5,1}=\bar{x}_1x_4$ $D_{5,4}=\bar{x}_4 \vee x_1x_4$
s_6	s_1	$D_{6,1}=1$

Таблица 2. Операции блока *alu*

Состояние	Операция	Тип операции
s_1	z = *a* and *b*	Логическая
s_2	z = *a* or *b*	
s_3	z = *a* xor *b*	
s_4	z = *a* xnor *b*	
s_5	z = *a* + *b*	Арифметическая
s_6	z = *a* · *b*	

Верификация на основе покрытия VHDL-кода

Понятие «покрытие» при верификации применяется в различных контекстах и связано с выполнением операторов программы либо использованием (покрытием) при моделировании возможных входных воздействий и получением тех или иных реакций. Покрытие может относиться также к переключениям сигналов. В этом случае переключение сигнала считается покрытым, если в процессе моделирования произошли переключения

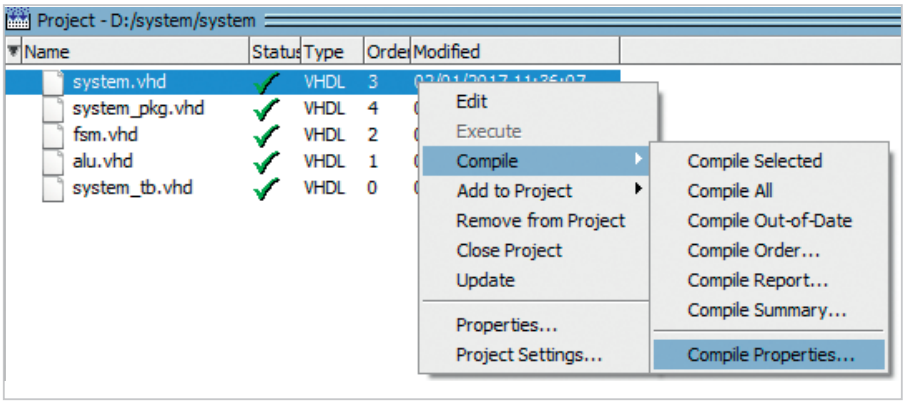

Рис. 3. Окно для установления опций компиляции

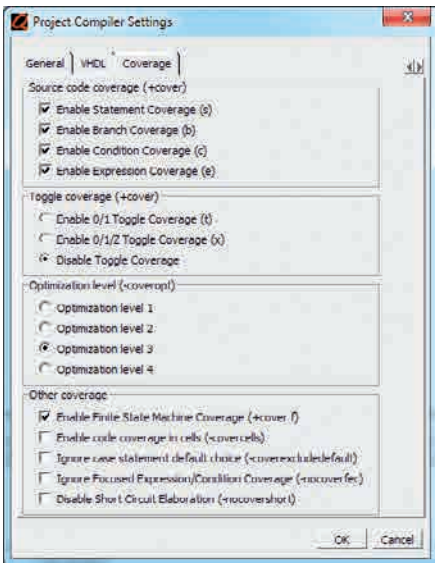

Рис. 4. Установление опций компиляции для покрытия кода

в «обе стороны», т.е. сигнал переключился из 0 в 1 и обратно – из 1 в 0. Кроме операторов анализу на покрытие могут подвергаться также строки VHDL-кода, выражения, экземпляры компонентов (операторы *port map*). Покрытие характеризует текст программы с точки зрения исполнения фрагментов VHDL-кода при моделировании.

Применение процедур покрытия кода не предназначено для проверки правильности ожидаемых и получаемых реакций VHDL-модели цифровой системы на наборах значений входных сигналов и не гарантирует полной верификации.

Покрытие VHDL-кода как вид верификации исходит из общих принципов проверки правильности программ, написанных на любом (необязательно на VHDL) языке программирования, и не гарантирует (даже при 100% покрытии) полной верификации проекта. Тем не менее такой вид проверки текстов VHDL-программ, представляющих собой модели устройств, позволяет находить достаточно много ошибок в описаниях функционирования либо в структурах устройств. Наиболее простым понятием покрытия кода является покрытие строк. Данный вид покрытия может относиться как к структурному, так и функциональному описаниям системы и определяется следующим образом: если в данном сеансе моделирования с данным набором тестирующих векторов конкретная строка VHDL-кода была выполнена хотя бы один раз, то она является покрытой, если же строка кода ни разу не выполнялась, то она является непокрытой. В одной строке кода может содержаться несколько исполняемых операторов, поэтому считается, что более содержательным является покрытие операторов (в данном случае вместо строки кода рассматривается покрытие каждого исполняемого оператора). Напомним, что декларации в VHDL не являются исполняемыми и не подвергаются анализу на покрытие. В системе Questa Sim при моделировании можно указать следующие опции покрытия кода:

- Enable Statement coverage – позволяет системе моделирования подсчитать число выполнений каждого оператора в строке.
- Enable Branch coverage – подсчитывается число выполнений условий типа *if/then/else* и *case* и определяются случаи, когда истинное или ложное условие не выполнилось.
- Enable Condition coverage – анализируются выборы, сделанные в условиях *if* и *case*; данная опция является расширением *Branch coverage*.
- Enable Expression coverage – анализирует выражение в правой части оператора назначения сигнала (присвоения значения переменной) аналогично *Condition coverage*.
- Enable 0/1 Toggle coverage – считаются переходы логического сигнала из одного состояния в другое; учитываются только переходы из 0 в 1 и обратно.
- Enable 0/1/Z Toggle coverage – считаются переходы логического сигнала из одного состояния в другое; учитываются только переходы между значениями сигнала из множества {0, 1, Z}, где Z – значение «высокий импеданс» сигнала типа *std_logic*.
- Enable State Machine coverage – в VHDL-коде выделяется описание конечного автомата, для которого при моделировании подсчитываются пройденные (покрытые) переходы между внутренними состояниями, в которые попадает автомат.

Маршрут моделирования с покрытием кода

Чтобы выполнить покрытие кода при моделировании, требуется установить соответствующие опции при компиляции и выполнении моделирования. Рассмотрим данный вид моделирования на примере цифрового устройства. Далее перечислены шаги для выполнения моделирования с покрытием кода. Предполагается, что проект создан и проверен с помощью обычного моделирования, т.е. без покрытия кода [5].

Шаг 1. Установка опций покрытия кода перед выполнением компиляции

В закладке Project сначала требуется отметить требуемые файлы проекта, для которых планируется анализировать покрытие кода, а затем по правой клавише мыши открыть окно Compile Properties (см. рис. 3). Если это один файл, то можно установить курсор на этом файле.

Открыв окно Compile Properties, в разделе Coverage требуется установить флаги (см. рис. 4):

- Enable Statement Coverage – для покрытия операторов;
- Enable Branch Coverage – для покрытия ветвлений;
- Enable Condition Coverage – для покрытия условных операторов;
- Enable Expression Coverage – для покрытия выражений;
- Enable Finite State Machine Coverage – для покрытия конечных автоматов.

После того как опции установлены, требуется нажать кнопку OK.

Шаг 2. Компиляция

Выполнить компиляцию можно различными способами: каждый модуль

может компилироваться по отдельности, либо все они компилируются по команде Compile All.

Шаг 3. Установка опций покрытия кода перед моделированием и выполнение моделирования

В закладке Simulation выбираем раздел Start Simulation, затем в закладке Others устанавливаем флаг Enable Code Coverage – «выполнять покрытие кода при моделировании» (см. рис. 5).

Затем осуществляется указание головного модуля, а после нажатия кнопки OK совершается переход в режим моделирования. Дальнейшие действия являются стандартными для выполнения моделирования с использованием окон графического интерфейса. В режиме моделирования из окон Object сигналы переносятся в окно Wave. Выполнение моделирования осуществляется по команде Run All.

Шаг 4. Анализ покрытия одной строки кода

Чтобы увидеть, какие строки кода конкретного модуля отмечены как покрытые (непокрытые), требуется открыть текст этого модуля. На рисунке 6 символами X_C, X_B, X_F отмечаются непокрытые строки VHDL-кода автомата. Значения данных символов, которые даны в отчёте о покрытии кода, приводятся в таблице 3.

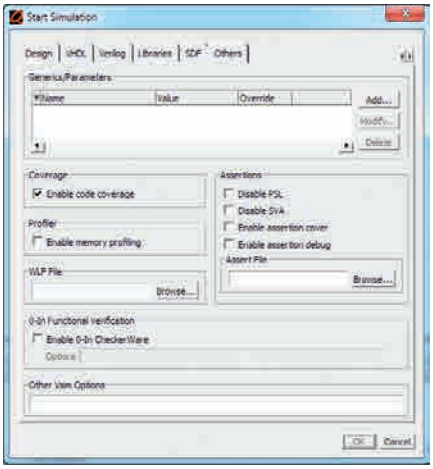

Рис. 5. Установка опции покрытия кода при установке опций моделирования

Установив курсор на строку, можно получить отчёт о покрытии данной строки. Отчёт о покрытии строки можно также получить, выполнив View → Coverage → Details.

Шаг 5. Формирование общего отчёта о покрытии VHDL-кода

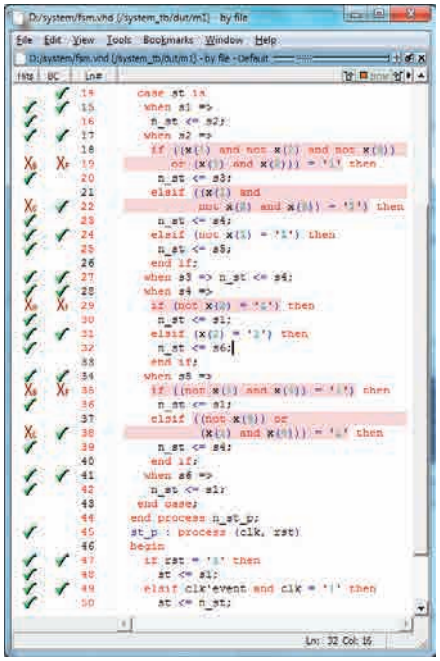

Рис. 6. Непокрытые строки VHDL-кода автомата *fsm*

Чтобы получить общий отчёт о покрытии кода, нужно выполнить Tools → Coverage Report и выбрать форму представления отчёта: в виде текстового файла (Text) либо HTML-

Таблица 3. Обозначение результатов покрытия VHDL-кода

Обозначение выполнения (невыполнения) покрытия	Интерпретация
	Все операторы, ветви, условия или выражения на строке кода были выполнены
X	Различные виды покрытия на строке кода не были выполнены
X_T	Ветви по значению «истина» (True) не были выполнены (пройдены)
X_F	Ветви по значению «ложь» (False) не были пройдены
X_C	Условия (Condition) не были выполнены
X_E	Выражения (Expression) не были выполнены
X_B	Ветвь (Branch) не была пройдена
X_S	Оператор (Statement) не был выполнен
E	Строки, которые исключаются при покрытии кода

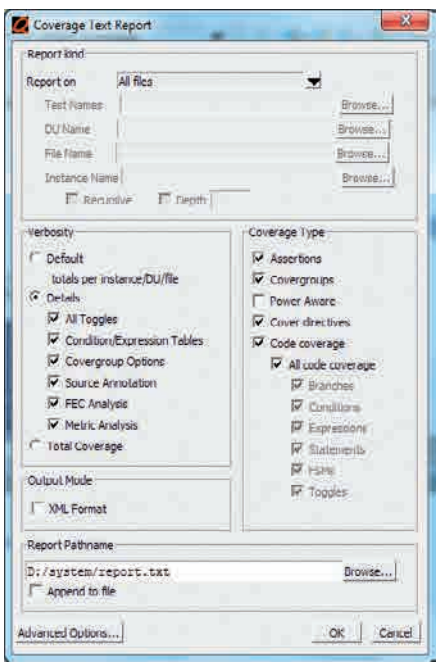

Рис. 7. Формирование текстового отчёта с указанием имени файла отчёта

файла (HTML). Выбрав Text, требуется в открывшемся окне указать имя файла текстового отчёта (на рисунке 7 этот файл имеет имя *report.txt)*.

Кроме текстовых файлов отчётов, удобно получить итоговый отчёт в виде HTML-файла. Например, выполнив моделирование устройства *system* на 10 000 псевдослучайных входных наборов, можно добиться следующих результатов покрытия VHDL-кода (см. рис. 8).

Total Coverage:					89.18%	86.15%
Coverage Type	Bins	Hits	Misses	Weight	% Hit	Coverage
Statements	19	19	0	1	100.00%	100.00%
Branches	26	22	4	1	84.61%	84.61%
FEC Conditions	10	6	4	1	60.00%	60.00%
FSMs	19	19	0	1	100.00%	100.00%
States	6	6	0	1	100.00%	100.00%
Transitions	13	13	0	1	100.00%	100.00%

Рис. 8. HTML-отчёт о полном покрытии кода

В файле *report.txt* даётся подробный анализ причин того, почему не покрыта соответствующая VHDL-конструкция в той или иной строке. В данном примере непокрытыми явились строки VHDL-кода, в которых записаны условия переходов, – это и привело к тому, что нет 100% покрытия ветвей и условий в управляющем автомате. В книге [6] подробно описываются наиболее распространённые причины отсутствия таких покрытий. Например, условные операторы *if* должны иметь часть *else*, логические условия переходов должны быть ортогональными (не пересекаться) и т.д.

Заключение

Верификация на основе покрытия VHDL-кода может быть выполнена для VHDL-описаний проекта в целом, а не только для VHDL-описаний цифровых устройств, представленных в виде соединения управляющего и операционного блоков. Она позволяет найти «мёртвые» участки кода, проанализировать полноту применяемых тестов для моделирования, пересечение (ортогональность) логических условий и т.д. Другие, более сложные проблемы верификации решаются на основе функциональной верификации, которая рассматривается во второй части данной статьи. Там же будут более подробно описаны специальные средства системы моделирования Questa Sim, предназначенные для верификации конечных автоматов.

Литература

1. *Иванюк А.А.* Проектирование встраиваемых цифровых устройств и систем. – Минск: Бестпринт, 2012.
2. *Поляков А.К.* Языки VHDL и VERILOG в проектировании цифровой аппаратуры. – М.: СОЛОН-Пресс, 2003.
3. *Чэнь М., Цинь К., Ку Х.-М., Мишра П.* Валидация на системном уровне. Высокоуровневое моделирование и управление тестированием. – М.: Техносфера, 2014.
4. *Лохов А., Рабоволюк А.* Комплексная функциональная верификация СБИС. Система Questa компании Mentor Graphics // Электроника: наука, технология, бизнес. 2007. №3. с. 102–109.
5. *Бибило П.Н.* Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. – М.: СОЛОН-Пресс, 2005.
6. *Бибило П.Н., Авдеев Н.А.* Моделирование и верификация цифровых систем на языке VHDL. – М.: ЛЕНАНД, 2017.

НОВОСТИ МИРА

«Микрон» ресертифицировал СМК по стандарту ISO 9001:2015

«Микрон» успешно прошёл ресертификацию системы менеджмента качества (СМК) по международному стандарту ISO 9001:2015. Аудит проведён международной независимой организацией Bureau Veritas Certification, ведущей свою историю с 1828 года.

Задача применения СМК на предприятии заключается в обеспечении конкурентоспособности выпускаемой продукции и повышении эффективности деятельности компании. На протяжении 20 лет «Микрон» поддерживает и совершенствует корпоративную политику соответствия международным стандартам качества. На сегодняшний день в компании действует программа внутренних аудитов и корректирующих действий, процессы планирования и оценки результатов, система мониторинга и контроля показателей реализации процессов СМК. Этот комплекс процессов обеспечивает планомерное сокращение производственного цикла и технологических потерь.

По результатам проведённого аудита принято положительное решение о выдаче «Микрону» сертификата Bureau Veritas Certification с аккредитацией UKAS (Великобритания).

www.mikron.ru

V МЕЖДУНАРОДНЫЙ ПРОМЫШЛЕННЫЙ ФОРУМ

НЕРАЗРУШАЮЩИЙ КОНТРОЛЬ
ИСПЫТАНИЯ • ДИАГНОСТИКА

ТЕРРИТОРИЯ NDT

27 ФЕВРАЛЯ • 1 МАРТА 2018
МОСКВА • ЦВК ЭКСПОЦЕНТР

WWW.EXPO.RONKTD.RU

РОНКТД

ОРГАНИЗАТОР:
РОССИЙСКОЕ ОБЩЕСТВО ПО НЕРАЗРУШАЮЩЕМУ КОНТРОЛЮ И ТЕХНИЧЕСКОЙ ДИАГНОСТИКЕ

НОВОСТИ МИРА

Итоги международного форума «Микроэлектроника 2017»

2–7 октября 2017 года в городе Алушта, Республика Крым, состоялся международный форум «Микроэлектроника 2017». Ежегодный форум проводился в третий раз при поддержке Департамента радиоэлектронной промышленности Минпромторга РФ, Госкорпорации «Ростех», Союза машиностроителей России, Министерства обороны РФ и ряда других ключевых структур отрасли.

Международный форум «Микроэлектроника» – это независимая высокоинтеллектуальная площадка для ведения конструктивного диалога между производственными объединениями, научным сообществом и представителями бизнес-структур микроэлектронного кластера и смежных высокотехнологичных отраслей.

Программа форума включила 8 секций научной конференции, среди которых «Навигационно-связные СБИС и модули», «Высокопроизводительные вычислительные системы», «Информационно-управляющие системы», «Технологии и компоненты микро- и наноэлектроники», «Изделия микроэлектроники общего и специального назначения», «Методы и алгоритмы САПР СБИС», «СВЧ интегральные схемы и модули», «Микросистемы». Помимо научных секций, состоялись заседания 8 круглых столов деловой программы, а также ставший традиционным конкурс стартапов «Фестиваль инноваций».

Всего на форуме было представлено 186 докладов, охвативших как наиболее актуальные темы общеотраслевого характера, так и важнейшие для радио- и микроэлектроники вопросы, связанные с разработкой профильного программного обеспечения, промышленных микросхем, космической электроники, навигационных систем, средств автоматизированного проектирования и других приоритетных технологических направлений в микроэлектронике.

В работе форума приняли участие более 400 специалистов радиоэлектронной отрасли, в том числе представляющие Российскую академию наук и ведущие вузы страны. На полях мероприятия удалось консолидировать более 178 системообразующих для отрасли предприятий и образовательных учреждений из 34 городов России, а также Республики Беларусь, Республики Армения, Китайской Народной Республики, что, безусловно, закрепило за мероприятием статус главного события года в области микроэлектроники в России.

Одной из основных тем деловой программы форума стало развитие цифровой экономики России и радиоэлектронной промышленности как неотъемлемой части реализации вектора, обозначенного Президентом Российской Федерации в своём послании к профильным ведомствам, на пути перехода к новому технологическому укладу.

Всесторонне эксперты обсудили вопрос диверсификации оборонно-промышленных предприятий, занятых в области производства микроэлектронной продукции и компонентной базы. Отдельно были рассмотрены вопросы разработки и вывода конкурентоспособной гражданской продукции мирового уровня на новые рынки, в том числе за счёт развития кооперации с производителями гражданской номенклатуры. Участники форума были единодушны в оценке проблемного поля, формирующего барьеры для развития микроэлектронной промышленности. В фокусе обсуждения оказались вопросы кадрового обеспечение предприятий, высокого уровня зависимости отрасли от программных решений и комплектующих зарубежного производства, несовершенства законодательной базы, а также необходимости модернизации ряда программных документов, формулирующих стратегию развития микроэлектронной промышленности, в условиях динамично меняющейся политической и экономической мировой конъюнктуры.

Завершил деловую программу форума конкурс «Фестиваль инноваций», который был организован АО «НИИМА «Прогресс» и Федеральной программой «Работай в России», при поддержке АО «Росэлектроника», а также инновационного центра «Сколково». Фестиваль стал объединяющей площадкой научной конференции и деловой программы мероприятия. Руководители высокотехнологичных стартапов презентовали свои разработки и перспективные решения в сфере микроэлектроники перед представителями инвестиционного и академического сообществ, а также производственных компаний. Высокотехнологичным компаниям, представлявшим свои проекты в финале конкурса, был присвоен статус «Технологический партнёр АО «Росэлектроника», который даёт возможность развития взаимовыгодного сотрудничества между финалистами и холдингом.

«Фестиваль инноваций» проходил во второй раз и вызвал большой интерес у делегатов форума, продемонстрировав высокий потенциал взаимодействия инновационных проектов молодых компаний и крупных предприятий отрасли.

Итоги мероприятия были подведены на заключительном круглом столе форума с участием руководителей секций научной конференции, ведущих конструкторов, профессоров и специалистов микроэлектронного кластера.

www.mri-progress.ru

Anritsu анонсировала новое ПО для анализатора устройств беспроводной связи MT8862A

Корпорация Anritsu анонсировала новое программное обеспечение для своего анализатора устройств беспроводной связи MT8862A Wireless Connectivity Test Set. Новая разработка, предназначенная для быстрорастущего рынка Интернета вещей (IoT), позволяет оценивать состояние устройств даже в том случае, когда используются средства защиты беспроводной сети.

Стандарты WLAN находят всё более широкое применение в современной технике: телевизорах, автомобилях, промышленном оборудовании и датчиках. Для обеспечения стабильного функционирования необходим контроль таких параметров, как чувствительность и зона покрытия сети, в условиях реальной эксплуатации.

Новое программное обеспечение MX886200A-020 позволяет Anritsu MT8862A использовать режим измерений Network Mode для тестирования работающих в беспроводной сети устройств, даже если для них активированы средства обеспечения безопасности. Программное обеспечение поддерживает ряд стандартов, включая WEP, WPA-Personal и WPA2-Personal.

Новая функциональность может быть добавлена в существующие приборы MT8862A посредством обновления встроенного программного обеспечения и установки лицензии через веб-браузер. Это избавляет от необходимости выполнять модернизацию в заводских условиях и сокращает время простоя до минимума.

MT8862A – это комплексный измерительный прибор для тестирования беспроводных устройств стандартов IEEE802.11ac/n/g/b/a. Система использует встроенные протоколы связи и обеспечивает тестирование таких характеристик беспроводных устройств, как РЧ-параметры приёмопередатчика (RF TRx), включая мощность передатчика (Tx), качество модуляции и чувствительность приёмника (Rx).

www.prist.ru

Электроника Транспорт 2018

12-я специализированная выставка электроники и информационных технологий для пассажирского транспорта и транспортной инфраструктуры

16-17 МАЯ, МОСКВА
КВЦ «СОКОЛЬНИКИ»
WWW.E-TRANSPORT.RU