УДК 681.326:519.613

**В.И. ХАХАНОВ, О.А. ГУЗЬ, И.А. ПОБЕЖЕНКО, NGENE CHRISTOPHER UMERAH**

***Харьковский национальный университет радиоэлектроники, Украина***

# ТЕХНОЛОГИЯ ТЕСТИРОВАНИЯ И ВЕРИФИКАЦИИ СИСТЕМНЫХ HDL-МОДЕЛЕЙ

*Технология позволяет осуществлять поиск ошибок с заданной глубиной в программном HDL-коде за приемлемое для разработчика время путем введения* ***ассерционной*** *избыточности в критические точки программной модели, определяемые с помощью синтезированных логических функций тестопригодности. Рассмотрены инновационные технологии тестопригодного проектирования программных и аппаратных продуктов, ориентированные на эффективную разработку тестов и верификацию компонентов цифровых систем на кристаллах. Таким образом, используемые в hardware design and test критерии управляемости и наблюдаемости применены для оценки качества программного кода в целях его улучшения и эффективного диагностирования семантических ошибок.*

***Ключевые слова:*** *тестирование, тестопригодность, верификация, ассерция, HDL-модель.*

## Введение

Цель – улучшение технологии тестирования и верификации цифровых систем для диагностирования и исправления ошибок HDL-моделей путем совместного использования механизма ассерций и технологий тестопригодного проектирования.

Задачи исследования:

1. Классификация технологий тестопригодного проектирования HDL-моделей для создания цифровых систем на кристаллах.

2. Разработка модели верификации и тестирования системной HDL-модели на основе использования ассерций.

3. Разработка метрики оценивания тестопригодности HDL-моделей на основе новой логической функции тестопригодности.

4. Применение технологической модели ассерций для верификации IP-core фильтра на основе дискретного косинусного преобразования.

5. Практические результаты и направления дальнейших исследований.

Источники исследования:

1. Технологии и средства создания тестов и testbench представлены в работах [1 – 3].

2. Модели и методы верификации системных моделей на основе механизма ассерций описаны в публикациях [4 – 7]. Тестопригодное проектирование программных продуктов использует стандарты IEEE [8 – 10], а также инновационные решения для верификации и анализа тестопригодности системных HDL-моделей [11 – 18].

## 1. Тестопригодность программно-аппаратных продуктов

Инициирующим ядром появления новых технологий тестирования и верификации в программной и компьютерной инженерии следует считать силиконовый кристалл, являющийся основой для создания вычислительных и/или коммуникационных устройств. Кристалл рассматривается как испытательный полигон для апробации новых средств и методов трассировки, размещения, синтеза и анализа компонентов. Технологические решения, выдержавшие испытания временем в микроэлектронике, далее захватываются и адаптируются в макроэлектронике, представленной компьютерными системами и сетями.

Сближение и взаимопроникновение технологий приводит к изоморфным методам проектирования, тестирования и верификации по отношению к программным и аппаратным комплексам, что по существу является закономерным процессом ассимиляции прогрессивных концепций. Тому способствует факт, что наиболее важные параметры жизненного цикла изделия, такие как time-to-market и yield становятся соизмеримыми по времени и выходу годной продукции. Кривая жизненного цикла аппаратного изделия, представленная на рис. 1, с точностью до изоморфизма отображает временные этапы программного продукта, который проходит аналогичные стадии: проектирование, увеличение объема выпуска и производство с доработкой и сопровождением изделия.

В контексте жизненного цикла существуют две актуальные проблемы, которые связаны с поднятием графика (кривой) вверх по оси ординат, а также с компрессией упомянутой кривой по временной оси, означающей уменьшение параметра time-to-market. Здесь повышение yield происходит на всех стадиях: design – за счет устранения ошибок разработки; production ramp up – за счет исправления кода, имплементированного в память системы на кристалле; volume – за счет выпуска service pack, корректирующих ошибки путем распространения через Internet или спутники (satellites).

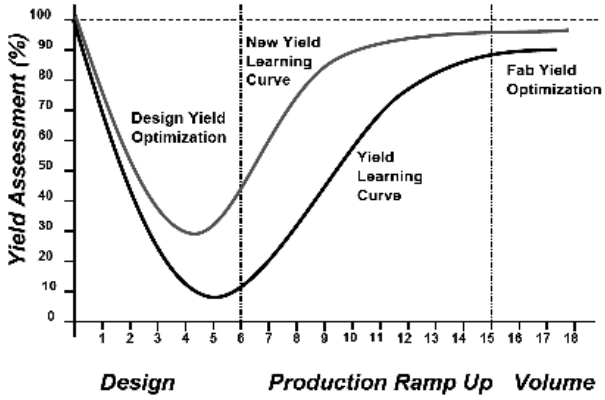


Рис. 1. График жизненного цикла программно-аппаратного комплекса

Согласно [13] стоимость верификации программно-аппаратных продуктов на основе ASIC, IP-core, SoC составляет 70% от общих затрат проектирования. Аналогичная оценка, около 80 %, определяет размерность testbench-кода от общей длины формального описания проекта. Задачи, решаемые в процессе верификации, связаны с устранением ошибок проектирования как можно на более ранней его стадии в целях приведения кода прототипа в соответствие с его спецификацией. Пропуск ошибки увеличивает ее стоимость на порядок при переходе от уровня проектирования блока до уровня кристалла и далее к системе.

Введение в проект программной избыточности – механизма ассерций [15] – позволяет выполнять анализ основных специфицированных условий в процессе моделирования проекта и диагностировать ошибки в случае их обнаружения на ранних стадиях проектирования программы (C++) или аппаратуры (HDL).

## 2. Инфраструктура процесса верификации проекта

Модель процесса верификации проекта на системном уровне можно представить в виде обобщенного уравнения обнаружения ошибок $T \oplus S = L$ или более подробно в компонентах:

$$(T, A) \oplus (P, S, F) = (L_V).$$

Здесь: $T, A$ – тестовые и ассерционные воздействия с ожидаемыми реакциями; $P, S, F$ – спецификация, HDL-модель функциональное покрытие оценки полноты теста для верификации проектируемого изделия. При тестировании аппаратной реализации вступает в силу аналитическое выражение $(T) \oplus (S, B) = (L_T)$, где B – регистр граничного сканирования от стандарта IEEE 1500, используемый в качестве дополнения к модели для обеспечения требуемой глубины диагностирования. При этом $L_V, L_T$ – списки ошибок и неисправностей, полученные на стадиях верификации проекта и тестирования готового изделия.

Для более точного понимания соотношений между ключевыми понятиями верификация, валидация, ассерция вводятся следующие определения [13, 14]. Верификация – есть процесс анализа системы или компонентов для определения корректности преобразований входного описания на очередной стадии проектирования. Валидация – есть процесс определения работоспособности системы и ее компонентов путем проверки соответствия основным требованиям спецификации после выполнения каждой стадии проектирования. Сертификация – есть письменная гарантия того, что система или ее компоненты полностью удовлетворяют требованиям спецификации и приемлемы для использования по назначению. Ассерция – есть высказывание системного уровня, определяющее корректность преобразований в процессе проектирования относительно входного описания текущего этапа или требований спецификации. Следуя последнему определению ассерцию можно записать в виде предиката $A_i = f(A_{i1}, A_{i2}, ..., A_{ij}, ..., A_{i_n}) = Y = \{0,1\}$, который на множестве переменных $i_n$ принимает значения {true, false}. В соответствии с упомянутым определением ассерции можно дифференцировать на три типа $A = \{A_f, A_p, A_s\}$ – проверки неисправностей, функциональных путей и возможных состояний проектируемого изделия. Такая классификация позволяет проверить в программном коде, как механизм вычислений, так и управления вычислительными процессами. Механизм ассерций – есть полная система высказываний $A = (A_1, A_2, ..., A_i, ..., A_n)$ и средства их анализа, предназначенные для верификации и валидации процесса проектирования, а также диагностирования ошибок.

Стратегии верификации и тестирования имеют различные модели приложения технологий, ориентированные на упомянутые процессы. Для верификации характерным признаком является работа

с HDL-моделью, полученной по спецификации проекта, которая проходит стадии преобразований, связанных с его проектированием (рис. 2) (синтез и имплементация) в силиконовый кристалл. Процесс тестирования здесь отображает взаимодействие аппаратной и HDL-модели на несистемном (регистровом или вентильном) уровне проектирования при имплементации идеи в кристалл программируемой логики. Для кристаллов ASIC такая технология нецелесообразна, поскольку перепрограммирование ошибки здесь будет стоить до миллиона долларов.

С учетом приведенных определений и пояснений модель среды или макропроцесса верификации программной стадии проекта ориентирована на уменьшение времени создания изделия и повышение уровня выхода годной продукции за счет использования избыточности кода в виде механизма ассерций и использования Testbench совместно с метрикой определения качества теста или функциональной полноты.

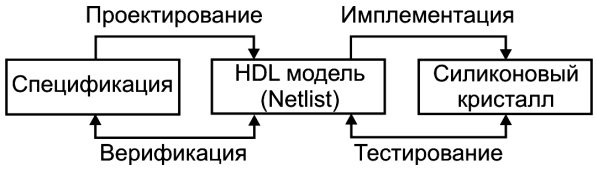


Рис. 2. Стратегия разработки проекта

Технология тестирования и верификации HDL-модели представлена на рис. 3, где спецификация проекта, описанная на формальном языке высокого уровня, является исходной информацией для: создания метрики оценивания качества теста, модели проекта, теста с эталонными реакциями – testbench, ассерционной структуры, служащей дополнением к основной модели, что необходимо для ускорения тестирования и отладки проекта.

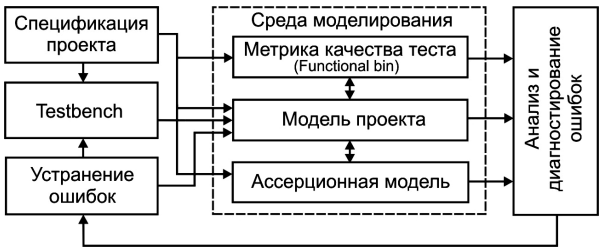


Рис. 3. Среда верификации проекта

Среда верификации представлена системой моделирования (например, Questa, Mentor Graphics), тестирования (Testbench), механизмом ассерций (Assertion Engine) и собственно системным кодом модели на языках VHDL, Verilog, System Verilog. Модуль Testbench задает входные стимулы и эталонные реакции на них, записанные на HDL-языках, ориентированные на проверку функциональностей (переменные, функции, последовательности), параметры которых определяются в функциональной корзине. Механизм ассерций – модельная избыточность, дополняющая testbench, в части проверки внутренних по времени состояний проекта, представленная вход-выходными высказываниями и предназначенная для ускорения тестирования, верификации, диагностирования и исправления ошибок проектирования в системном коде. Ассерции можно сгенерировать не по спецификации, но и по HDL-модели, убирая ненужные конструкции, а остальные необходимо модифицировать к форме ассерций.

## 3. Анализ тестопригодности программных HDL-моделей

Для идентификации обобщенного состояния HDL-модели формируются эталонные сигнатуры критических точек во времени и в пространстве. Затем выполняется диагностирование HDL-модели по сигнатурам ассерционной матрицы, описывающей состояние проекта во времени и в пространстве. Целесообразно формировать ассерционную матрицу независимо от HDL-модели проекта. Ассерционная и функциональная модели обрабатываются параллельно и независимо друг от друга средой моделирования. Ассерционная модель ориентирована на тестирование внутренних во времени и в пространстве точек проекта, обозначенных его пространственно-временной матрицей. Ассерционная модель определяет поведение проекта в существенных точках пространственно-временной эталонной матрицы. Формат ассерций должен соответствовать формату матрицы HDL-модели проекта. Ассерции ориентированы на дополнение полноты Testbench в части тестирования внутренних точек матрицы HDL-модели. Аналитическая форма представления инфраструктуры верификации представлена следующими выражениями (P – спецификация проекта, S – soft-модель, A – ассерционная модель, T – Testbench, F – корзина задания функциональностей для определения их полноты покрытия, D – модуль диагностирования ошибок и C – условия диагностирования ошибок):

$$M = \{P, S, A, T, F, D, C\},$$

$$1)\ S = f_1(P) = \left|S_{ij}\right|;$$

$$2)\ F = f_2(P, S) = \{F_1, F_2, ..., F_i, ..., F_n\};$$

$$3)\ T = f_3(P, S, F) = \{T_1, T_2, ..., T_i, ..., T_n\};$$

$$4)\ A = f_4(P, S, F, T) = \left|A_{ij}\right|;$$

$$5)\ D = f_5(P, S, F, T, A) = \left|L_{ij}\right| \in [L^V \vee L^T];$$

$$6)\ C = [\bigcup_{i=1}^{n} F_i \in F = P] \wedge [\bigcup_{i=1}^{n} T_i \in T = F];$$

$$7)\ L^V = (T \oplus P) \vee (A = T^A \oplus P^A);$$

$$8)\ L^T = (T \oplus S).$$

Здесь выражение 6 определяет условия полноты функциональных корзин относительно спецификации и полноты теста относительно функциональностей проекта. Строка 7 задает функцию вычисления ошибок проектирования при переходе от системного к регистровому уровню, используя все атрибуты верификационной инфраструктуры. Функция 8 регламентирует нахождение неисправностей на стадии эксплуатации цифровой системы на кристалле. Следует отметить, что ассерционная избыточность является функцией от критических точек функциональной модели, предельное число которым может быть равно числу, определенных спецификацией, временных фреймов функциональных компонентов, что иллюстрируется следующими взаимодействующими матрицами ($T_i^B, T_i^A, S_i^t, A_i^t, R_i^B, R_i^A$ – testbench, ассерционные входные стимулы, функциональный компонент в момент t, ассерционный компонент в момент t, реакция функции, реакция ассерции):

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $T_1^B$ | Sim → | $S_1^1$ | $S_1^2$ | $S_1^t$ | $S_1^m$ | Sim → | $R_1^B$ |
| $T_2^B$ | | $S_2^1$ | $S_2^2$ | $S_2^t$ | $S_2^m$ | | $R_2^B$ |
| $T_i^B$ | | $S_1^1$ | $S_i^2$ | $S_i^t$ | $S_i^m$ | | $R_i^B$ |
| $T_n^B$ | | $S_n^1$ | $S_n^2$ | $S_n^t$ | $S_n^m$ | | $R_n^B$ |
| | | ↕ | | | | | |
| $T_1^A$ | Sim → | $A_1^1$ | $A_1^2$ | $A_1^t$ | $A_1^m$ | Sim → | $R_1^A$ |
| $T_2^A$ | | $A_2^1$ | $A_2^2$ | $A_2^t$ | $A_2^m$ | | $R_2^A$ |
| $T_i^A$ | | $A_1^1$ | $A_i^2$ | $A_i^t$ | $A_i^m$ | | $R_i^A$ |
| $T_n^A$ | | $A_n^1$ | $A_n^2$ | $A_n^t$ | $A_n^m$ | | $R_n^A$ |

Матрица ассерций должна создаваться независимо от матрицы функциональностей и желательно другим разработчиком. Такое условие обеспечит диверсификацию упомянутых моделей относительно единой спецификации, а также обнаружение и исправление ошибок в обеих матрицах.

Интерес представляет и последовательность действий по созданию среды верификации, где единственным аргументом является спецификация проекта, все остальное – производные от нее, представленные матрицей контрдостижимостей:

$$M = \begin{array}{|c|c|c|c|c|c|} \hline P & S & & & & \\ \hline P & S & F & & & \\ \hline P & S & F & T & & \\ \hline P & S & F & T & A & \\ \hline P & S & F & T & A & D \\ \hline \end{array}$$

В соответствии с моделью M на рис. 4 изображена структура взаимосвязей процесса проектирования и диагностирования с последующим исправлением ошибок в HDL-коде программы.

Достаточно существенная избыточность HDL-модели предполагает определение эффективности ее использования в целях повышения тестопригодности структуры разработанного или написанного кода.

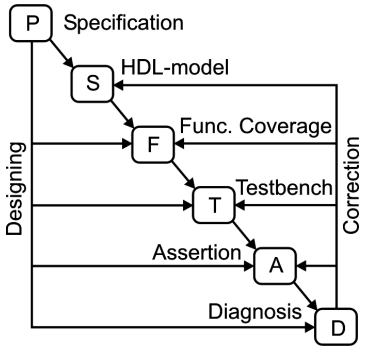


Рис. 4. Маршрут верификации проекта

Используя стандарты тестопригодного проектирования аппаратуры, можно перенести и адаптировать основные понятия применительно к HDL-коду системных и регистровых программных моделей. Здесь можно использовать граф регистровых или транзакционных передач С.Г. Шаршунова, который предоставляет пользователю информацию о взаимосвязях булевых и регистровых переменных, памяти и интерфейсных шинах. Такие данные достаточно легко можно получить автоматически, анализируя синтаксис строк HDL-кода. Сгенерированный граф должен покрывать упомянутые выше компоненты программной модели и обозначать все существующие связи для приема, передачи и преобразования информации между вершинами графа транзакций (TG – Transaction Graph).

Управляемость и наблюдаемость есть метрика оценивания тестопригодности не соединительных линий, а компонентов HDL-кода, таких как: регистр, счетчик, память или массивы, вход-выходные шины, векторы, логические или арифметические переменные. Указанные выше характеристики имеют функциональную зависимость от структурной глубины нахождения компонента (в пространстве или во времени), а также от процентного отношения числа команд, имеющих входной (выходной) доступ к вершине при анализе данной программы:

$$Q = \frac{1}{Z}(U \times N) = \frac{Z(S)}{Z(S) + Z(F) + Z(T) + Z(A)} \times$$
$$\times (\frac{1}{n}\sum_{i=1}^{n} U_i) \times (\frac{1}{n}\sum_{i=1}^{n} N_i);$$
$$U_i = \frac{1}{T}\sum_{j=1}^{x_i} T_j^i \times \frac{1}{d_i^x \vee t_i^x}; \quad (1)$$
$$N_i = \frac{1}{T}\sum_{j=1}^{y_i} T_j^i \times \frac{1}{d_i^y \vee t_i^y}.$$

Тестопригодность Q, представленная в (1), зависит от управляемости U, наблюдаемости (N), а также от модельной избыточности, представленной компонентами: метрика функционального покрытия (F), testbench (T), механизм ассерций (A). При этом управляемость (наблюдаемость) есть функция от числа операторов, входящих в вершину (исходящих из вершины) транзакционного графа, а также от структурной глубины рассматриваемого элемента – расстояния от входной (выходной) шины или от количества временных тактов, необходимых для управления (наблюдения) компонента в заданном состоянии на временной оси. Приведенный критерий тестопригодности может быть также использован и для оценки качества граф-схемы управления вычислительным процессом. Здесь рассматриваются только операторные вершины, нагруженные входными условиями, а также позиция вершины по отношению к началу или окончанию схемы управления. Позиция операторной вершины коррелируется с временным тактом управления вычислительным процессом. Количество условий выполнения совокупности операций в каждой вершине, объединенное операцией Or, повышает тестопригодность графа в части управляемости. Аналогично вычисляется наблюдаемость, на которую влияет не только структурная глубина, но и мощность условий, создаваемая функциональными операциями And, Or. Структурная глубина рассматриваемой вершины, а значит и тестопригодность может быть опосредована только логической функцией, заданной в виде конъюнктивной нормальной формы КНФ. При этом тестопригодность (управляемость и наблюдаемость) будет определяться оценкой по Квайну вычислительной сложности КНФ. В общем случае логические функции управляемости и наблюдаемости текущей вершины транзакционного графа представлены конъюнкцией дизъюнктивных термов:

$$U_i = \frac{1}{T}(\bigwedge_{j=1}^{x_1} T_{1j}^x)...(\bigwedge_{j=1}^{x_{i-1}} T_{i-1j}^x)(\bigvee_{i=1}^{x_i} T_{ij}^x);$$
$$N_i = \frac{1}{T}\bigwedge_{j=1}^{x_i} T_j^i(\bigvee_{i=1}^{x_i} T_j^i d_i^x \vee t_i^x). \quad (2)$$

Здесь мощность дизъюнктивных термов соответствует количеству входящих в вершину дуг, а число конъюнкций есть структурная глубина местоположения рассматриваемого компонента.

Интересным представляется решение, когда управляемость и наблюдаемость текущей вершины транзакционного графа вычисляется на основании построенных логических функций управляемости и наблюдаемости ( $U_i, N_i$ ) при использовании аппарата – алгебраической формы представления графа [18]. Формулы подсчета упомянутых критериев имеют следующий вид:

$$U_i = \frac{1}{t_{max}^x \times n_t^x} \times \sum_{i=1}^{n_t^x}\sum_{j=1}^{k_i^x}(t_{max}^x - \left|t_{ij}^x\right| + 1);$$
$$N_i = \frac{1}{t_{max}^y \times n_t^y} \times \sum_{i=1}^{n_t^y}\sum_{j=1}^{k_i^y}(t_{max}^y - \left|t_{ij}^y\right| + 1), \quad (3)$$

где $t_{max}^x, n_t^x, k_i^x, \left|t_{ij}^x\right|$ – для критерия управляемости конъюнктивный терм максимальной длины; количество термов в логической функции управляемости; количество транзакций (букв) в текущем терме функции; мощность рассматриваемой транзакции в терме. Аналогичные обозначения используются и для критерия наблюдаемости – $t_{max}^y, n_t^y, k_i^y, \left|t_{ij}^y\right|$.

Для фрагмента графа, представленного на рис. 5, преобразование конъюнктивной формы в дизъюнктивную структуру для вершины $V_3$ по выражению (2) формирует логическую функцию управляемости:

$$V_3 = (T_1 \vee T_2 \vee T_3)T_5T_8 \vee (T_4 \vee T_6)T_8 \vee (T_7 \vee T_9) =$$
$$T_1T_5T_8 \vee T_2T_5T_8 \vee T_3T_5T_8 \vee T_4T_8 \vee T_6T_8 \vee T_7 \vee T_9.$$
$$V2 = (T_1 \vee T_2 \vee T_3)T_5 \vee T_4 \vee T_6 =$$
$$= T_1T_5 \vee T_2T_5 \vee T_3T_5 \vee T_4 \vee T_6.$$
$$V_1 = T_1 \vee T_2 \vee T_3.$$

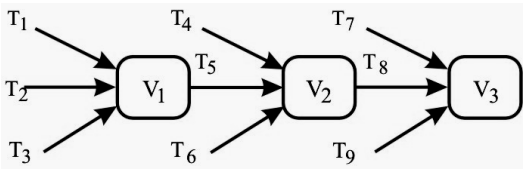


Рис. 5. Фрагмент графа для подсчета управляемости

Следуя правилам, представленным в формуле (3), осуществляется определение управляемости компонента $V_3$:

$$U_i = \frac{1}{t_{max}^x \times n_t^x} \times \sum_{i=1}^{n_t^x}\sum_{j=1}^{k_i^x}(t_{max}^x - \left|t_{ij}^x\right| + 1) =$$
$$= \frac{1}{3 \times 7} \times (1+1+1+2+2+3+3) = 0{,}61$$

Для другого фрагмента графа, представленного на рис. 6, преобразование конъюнктивной формы в дизъюнктивную структуру позволяет определить логическую функцию наблюдаемости вершины $V_1$:

$$V_1 = T_6 \vee T_7(T_3 \vee T_5 \vee T_4(T_1 \vee T_2)) =$$
$$= T_6 \vee T_7T_3 \vee T_7T_5 \vee T_7T_4T_1 \vee T_7T_4T_2;$$
$$V_2 = T_3 \vee T_5 \vee T_4(T_1 \vee T_2) = T_3 \vee T_5 \vee T_4T_1 \vee T_4T_2;$$
$$V_3 = T_1 \vee T_2 .$$

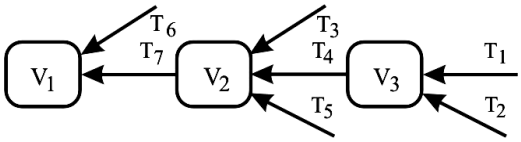


Рис. 6. Фрагмент графа для подсчета наблюдаемости

$$N_i = \frac{1}{t^y_{max} \times n^y_t} \times \sum_{i=1}^{n^y_t} \sum_{j=1}^{k^y_i} (t^y_{max} - \left|t^y_{ij}\right| + 1) =$$

$$= \frac{1}{3 \times 5} \times (3 + 2 + 2 + 1 + 1) = \frac{9}{15} = 0{,}6$$

Для случая, когда дуги в транзакционном графе имеют весовые коэффициенты, показывающие число операторов, задействованных в передаче информации между вершинами (компонентами), формулы подсчета тестопригодности имеют следующий вид:

$$U_i = \frac{1}{t^x_{max} \times (n^x_t = \sum_{i=1}^{n^x_t} \prod_{j=1}^{k^x_i} b^x_{ij})} \times$$

$$\times \sum_{i=1}^{n^x_t} \prod_{j=1}^{k^x_i} b^x_{ij} (t^x_{max} - \left|t^x_{ij}\right| + 1);$$

$$N_i = \frac{1}{t^y_{max} \times n^y_t = (\sum_{i=1}^{n^y_t} \prod_{j=1}^{k^y_i} b^y_{ij})} \times$$

$$\times \sum_{i=1}^{n^y_t} \prod_{j=1}^{k^y_i} b^y_{ij} (t^y_{max} - \left|t^y_{ij}\right| + 1).$$

## Заключение

Рассмотрены инновационные технологии тестопригодного проектирования программных и аппаратных продуктов [1 – 17], ориентированные на эффективную разработку тестов и верификацию компонентов цифровых систем на кристаллах.

1. Показаны основные направления использования технологий тестопригодного проектирования цифровых систем на кристаллах в задачах тестирования и верификации программных продуктов.

2. Представлена универсальная модель программного компонента в виде транзакционного графа, на котором можно решать задачи анализа тестопригодности в целях достижения требуемой глубины диагностирования HDL-кода.

3. Предложены логические функции тестопригодности HDL-моделей на основе использования транзакционного графа в целях вычисления оценок тестопригодности (управляемость и наблюдаемость) программных компонентов и всего HDL-проекта в целом.

4. Приведены примеры и графики оценивания тестопригодности (управляемости и наблюдаемости) программных моделей, представленных фрагментами транзакционных графов.

## Литература

*1. Abramovici M. Digital System Testing and Testable Design. Computer Science Press / M. Abramovici, M.A. Breuer, A.D. Friedman .– 1998.– 652 p.*

*2. Bayraktaroglu I. The Construction of Optimal Deterministic Partitionings in Scan-Based BIST Fault Diagnosis: Mathematical Foundations and Cost-Effective Implementations / I. Bayraktaroglu, A. Orailoglu // IEEE Transactions on Computers. – 2005.– P.61-75.*

*3. Douglas D. A Platform-Based taxonomy for ESL Design / D. Douglas, P. Roberto, S.V. Alberto // Design & Test of computers. – September-October, 2006. – P. 359-373.*

*4. Francisco D. Overview of the IEEE P1500 Standard / D. Francisco, Z. Yervant, W. Lee, A. Karim, K. Rohit // ITC International Test Conference. – 2003. – P. 988-997.*

*5. Rashinkar P. System-on-chip Verification: Methodology and Techniques / P. Rashinkar, P. Paterson, L. Singh. –Kluwer Academic Publishers. – 2002.– 324 p.*

*6. Zorian Y. What is Infrastructure IP? / Y. Zorian // IEEE Design & Test of Computers. – 2002.– P. 5-7.*

*7. Zorian Y. Guest editors' introduction: Design for Yield and reliability / Y. Zorian, D. Gizopoulos // IEEE Design & Test of Computers. – 2004. – P. 177-182.*

*8. Zorian Y. Guest Editor's Introduction: Advances in Infrastructure IP / Y. Zorian // IEEE Design and Test of Computers. – 2003. – 49 p.*

*9. Thatte S.M. Test generation for microprocessors / S.M. Thatte, J.A. Abraham // IEEE Trans. Comput.– 1980. – C-29. No 6. – P. 429-441.*

*10. Шаршунов С.Г. Построение тестов микропроцессоров. 1. Общая модель. Проверка обработки данных / С.Г. Шаршунов // Автоматика и телемеханика. – 1985. – №11.– С. 145-155.*

*11. Jerraya A.A. System Level Synthesis SLS. TIMA Laboratory. Annual Report / A.A. Jerraya. – 2002. – P. 65-75.*

*12. Frank G. Transaction Level Modeling with SystemC. TLM Concepts and Applications for Embedded Systems. / G. Frank // Published by Springer. – 2005. – 282 p.*

*13. Bergeron J. Writing testbenches: functional verification of HDL models. / J. Bergeron. – Boston: Kluwer Academic Publishers. – 2001. – 354 p.*

*14. Janick B. Verification Methodology. Manual for SystemVerilog / B. Janick, C. Eduard, H. Alan, N. Andrew. – Springer, 2005. – 528 p.*

*15. Harry F. Assertion-based design.– Second edition / F. Harry, K. Adam, L. David. – Kluwer Academic Publishers – Springer, 2005.– 392 p.*

*16. Rashinkar P. System-on-chip Verification: Methodology and Techniques / P. Rashinkar, P. Paterson, L. Singh. – Kluwer Academic Publishers, 2002. – 393 p.*

*17. Meyer A.S. Principles of Functional Verification. Elsevier Science, 2004. –206 p.*

*18. Хаханов В.И. Проектирование и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. – Х., 2009. – 484 с.*

*Поступила в редакцию 10.02.2010*

**Рецензент:** д-р техн. наук, проф., проф. кафедры Ф.В. Новиков, Харьковский национальный экономический университет, Украина.

## ТЕХНОЛОГІЯ ТЕСТУВАННЯ ТА ВЕРИФІКАЦІЇ СИСТЕМНИХ HDL-МОДЕЛЕЙ

***В.І. Хаханов, О.О. Гузь, І.О. Побіженко, Ngene Christopher Umerah***

Технологія дозволяє здійснювати пошук помилок із заданою глибиною в програмному HDL-коді за прийнятний для розробника час шляхом введення ассерціонної надмірності в критичні точки програмної моделі, що визначаються за допомогою синтезованих логічних функцій тестопридатності. Розглянуто інноваційні технології тестопридатного проектування програмних та апаратних продуктів, орієнтовані на ефективну розробку тестів і верифікацію компонентів цифрових систем на кристалах. Таким чином, використовувані в hardware design and test критерії спостереження та управління застосовані для оцінки якості програмного коду з метою його покращення та ефективного діагностування семантичних помилок.

**Ключові слова:** тестування, тестопридатність, верифікація, асерція, HDL-модель.

## TECHNOLOGY FOR TESTING AND VERIFICATION OF SYSTEM HDL-MODELS

***V.I. Hahanov, O.O. Guz, I.O. Pobizhenko, Ngene Christopher Umerah***

The technology allows you to search for bugs with a given depth in the program HDL-code within a reasonable time for the developer by introducing assertsionnoy redundancy in the critical points of the programming model, that are defined by the synthesized logic functions testability. Testability is considered to be innovative in technologies that design software and hardware products, targeting the effective development of test and verification components of digital systems in crystals. Thus, they are used in hardware design and in order to test the criteria for controllability and observability, they are applied to assess the quality of code in order to improve and effectively diagnose semantic errors.

**Keywords:** testing, testability, verification, assertion, HDL-model.

**Хаханов Владимир Иванович** – декан факультета КИУ ХНУРЭ, д-р техн. наук, проф. кафедры автоматизации проектирования вычислительной техники, Харьковский национальный университет радиоэлектроники, Харьков, Украина, e-mail: hahanov@kture.kharkov.ua.

**Гузь Олеся Алексеевна** – зав. кафедрой специализированных компьютерных систем, Донецкая академия автомобильного транспорта, Донецк, Украина, e-mail: hahanov@kture.kharkov.ua.

**Побеженко Ирина Александровна** – аспирантка кафедры автоматизации проектирования вычислительной техники, Харьковский национальный университет радиоэлектроники, Харьков, Украина, e-mail: hahanov@kture.kharkov.ua.

**Ngene Christopher Umerah –** аспирант кафедры автоматизации проектирования вычислительной техники, Харьковский национальный университет радиоэлектроники, Харьков, Украина, e-mail: hahanov@kture.kharkov.ua.