УДК 681.326:519.713

Е.И. ЛИТВИНОВА

ИНФРАСТРУКТУРА ВЕРИФИКАЦИИ И ТЕСТИРОВАНИЯ SOC

Предлагается инфраструктура тестирования и верификации цифровых систем на кристаллах, позволяющая оценивать тестопригодность программно-аппаратных модулей путем построения транзакционного графа для использования механизма ассерций и IEEE 1500 SECT стандарта, что дает возможность повысить эффективность сервисных средств моделирования, диагностирования и восстановления работоспособности, а также существенно уменьшить время верификации HDL-моделей и тестирования аппаратных компонентов SoC.

Задачи исследования: 1. Разработка инфраструктуры верификации, совместимой с современными стандартами и технологиями проектирования цифровых систем на кристаллах от ведущих компаний планеты. 2. Разработка аналитической модели оценивания технологических и структурных решений для анализа цифровых систем, совместимых по интерфейсу с продуктом моделирования QuestaSim для верификации и исправления ошибок в процессе создания HDL-модели. 3. Валидация программно-аппаратных компонентов инфраструктуры отладки HDL-кода, совместимой с существующими системами моделирования, в том числе использующих аппаратные акселераторы. 4. Оценка эффективности промышленного применения инфраструктуры тестирования и верификации на реальных примерах цифровых систем на кристаллах.

1. Инфраструктура проектирования и верификации SoC

Общая инфраструктура разработки и верификации цифровой системы на кристалле является инвариантной (универсальной) по отношению к прототипу и HDL-модели (рис. 1).

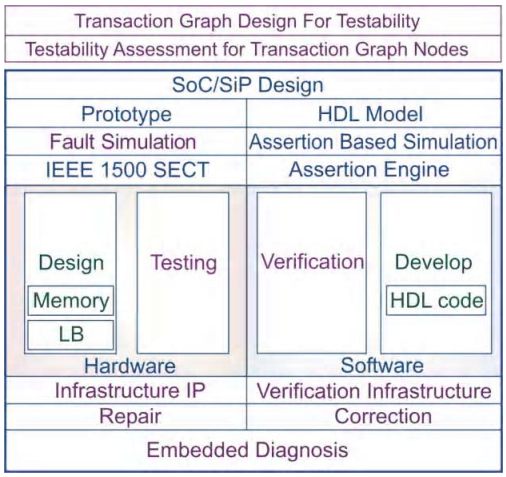

Рис. 1. Инфраструктура проектирования и верификации SoC

Она представляет собой концентрированную совокупность исследований и разработок [1-25], которая содержит следующие компоненты: 1. Transaction Graph Design for Testability [3], что представляет собой создание транзакционной модели программного кода, в которой вершинами являются логические и регистровые переменные, векторы, массивы и память, а дугами – операторы кода, выполняющие транзакции между вершинами. Транзакционный граф в части верификации прототипа строится и для аппаратных компонентов системы. 2. Testability Assessment for Transaction Graph Nodes [24-25] представляет собой вычислительные модели процессов оценки тестопригодности вершин транзакционного графа HDL-кода и прототипа. После этого выполняется процедура выбора достаточного количества вершин транзакционного графа с минимальной оценкой тестопригодности, в которые устанавливаются ассерции для HDL-модели или точки контроля для прототипа путем использования стандарта IEEE 1500 SECT. 3. SoC/SiP Design – выполняется синтез

основных функциональностей на основе применения существующих решений в виде IP-cores из библиотек ведущих компаний мира или путем разработки оригинального кода отдельных функциональностей (рис. 2).

Здесь исходной информацией является спецификация, конечным результатом – прототип, имплементированный в силиконовый кристалл. Кроме того, используется аппаратный акселератор компании Aldec [1-3], позволяющий в десятки раз повысить быстродействие моделирования. 4. Prototype – готовое аппаратное (программное) изделие, свободное от обнаруженных дефектов. 5. HDL-Model – результат автоматического генерирования HDL-кода из спецификации путем использования системных языков описания аппаратуры и промышленных симуляторов [4]. 6. IEEE 1500 SECT – стандарт, ориентированный на сервисное диагностическое обслуживание цифровых систем на кристаллах в процессе проектирования, производства и эксплуатации [5]. 7. Assertion Engine – технологический аппарат тестопригодного проектирования HDL-моделей цифровых систем, ориентированный на контроль исполнения программного кода в его критических точках [6]. 8. Design – проект цифровой системы на кристалле, ориентированный на выполнение актуальных для пользователя функциональностей, которые имплементированы в hardware. 9. Testing – процесс определения технического состояния изделия после его имплементации в силиконовый кристалл. 10. Verification – процесс определения технического состояния изделия на различных стадиях его проектирования путем сравнения со спецификацией на основе моделирования тестовых или функциональных воздействий. 11. Development – итеративный процесс создания изделия на основе спецификации с помощью промышленных систем моделирования и синтеза. 12. Hardware – аппаратная реализация функциональностей цифровой системы на кристалле, ориентированная на выполнение актуальных для EDA-рынка задач. 13. Software – программная реализация функциональностей цифр
ле. 14. Infrastructure IP – инфраструкт
вания функциональных компонентов
вания, производства и эксплуатаци
Infrastructure – совокупность про-
граммно-аппаратных средств, ори-
ентированная на проверку функци-
онирования и коррекции программ-
ного кода. 16. Repair – восстанов-
ление работоспособности аппарат-
ных компонентов системы на крис-
талле путем использования тесто-
вых или функциональных воздей-
ствий, процедур диагностирования
и средств наблюдения от стандар-
та IEEE 1500 SECT. 17. Correction
– процесс устранения семантичес-
ких ошибок HDL-кода, искажающих г
вание модели системы на кристалле
ции. 18. Embedded Diagnosis – встро
неисправностей HDL-модели или пр
внешних средств синтеза и анализа т

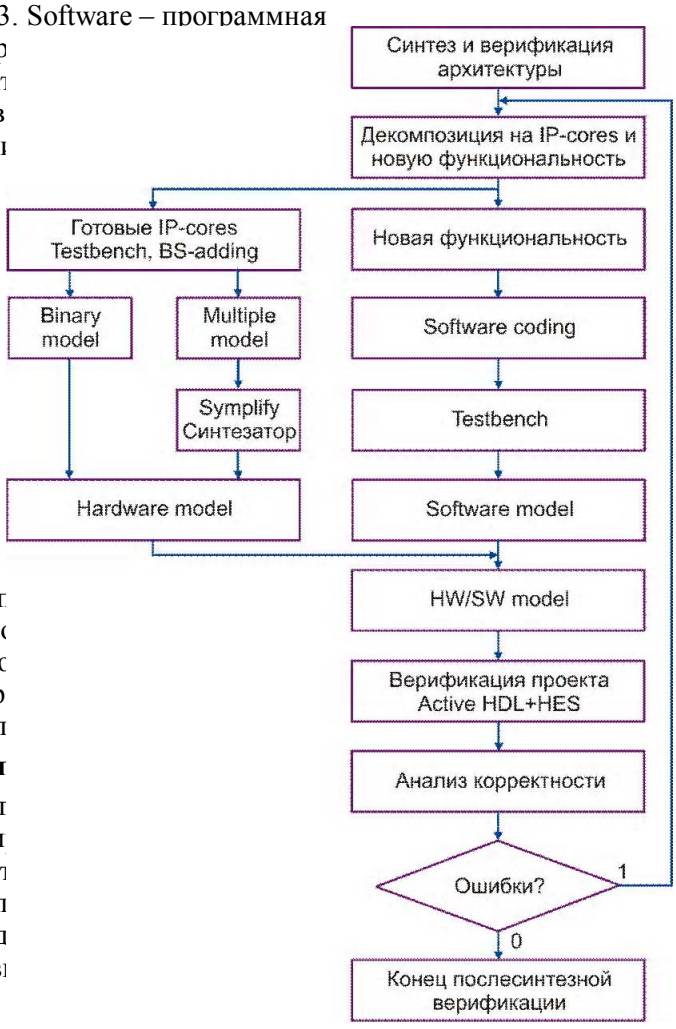

Рис. 2. Маршрут синтеза и верификации SoC

2. Верификация I-IP SoC комп

Процесс верификации компонент
висного обслуживания, как и проекти
этапов (рис. 3). Первый соответствует
нального поведения с помощью модел
скрипт программы Matlab для послед
го эталона, относительно которого в
последующих моделей. М-скрипт
позволяет считывать исходную ин-

формацию из файла, создавать тестовые и эталонные последовательности для модели устройства, выполнять анализ полученных результатов, формировать графический вывод исходных данных и результатов. Второй этап верификации соответствует модели системного уровня проектирования, с архитектурой исполнимой модели, разработанной в Simulink. При этом ввод тестов и анализ результатов осуществляется на основе скриптовых функций программной среды MATLAB, созданных на предыдущем этапе проектирования.

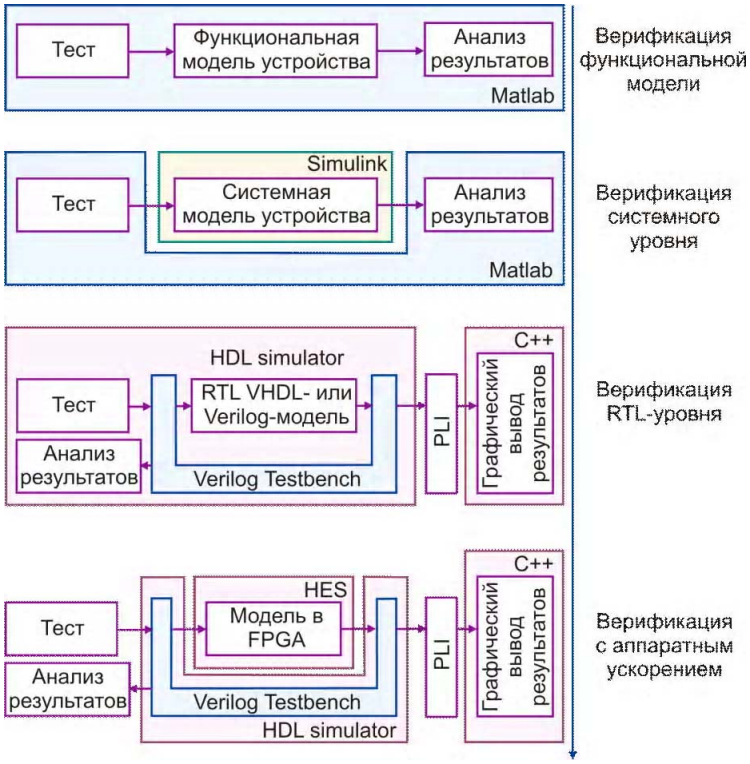

Рис. 3. Схема верификации компонентов I-IP SoC

Верификация RTL-модели, реализованной с помощью языков описания аппаратуры VHDL [10] или Verilog [11-14], выполняется с помощью HDL-симулятора, например Active-HDL или Modelsim. Testbench удобнее реализовать с использованием языка Verilog, поскольку он позволяет подключать через PLI-интерфейс программные модули, созданные с помощью языка C/C++ для визуального контроля процесса верификации. Для ускорения процессов тестирования и аппаратной верификации используется разработанная компанией Alatek плата аппаратного ускорения моделирования HES – Hardare Embedded Simulator. Компилирование проекта выполняется в Active-HDL симуляторе. Далее проект делится на 2 части: одна остается в симуляторе, а другая загружается в HES board. Моделирование проекта в аппаратуре выполняется на частоте 400 МГц. Testbench остается в программном симуляторе Active HDL.

Общая инфраструктура процесса тестирования на основе аппаратной верификации представлена на рис. 4. Здесь фигурируют компоненты: система моделирования Active HDL, Design Verification Manager, модуль создания аппаратной модели проекта и плата аппаратного ускорителя на основе кристалла Xilinx Virtex 4. В качестве примера следует привести основные параметры создания и отладки индустриально-ориентированного проекта Image Processing Design. Время подготовки модели – 40 часов. Время HW-моделирования – 25 секунд, SW-моделирования – 4 часа. Среднее ускорение моделирования для 5 промышленных проектов (900 – 5000 строк HDL-кода) равно двенадцать раз. Дополнительные средние затраты на создание аппаратной модели – 35 часов.

Эффективность разработанной инфраструктуры верификации и тестирования цифровых систем на кристаллах, а также быстродействие программных продуктов относительно базового варианта определяется по следующим формулам:

$$Q^t = \frac{1}{4}\left(\frac{Y^n}{Y^b}k_y + \frac{T^b}{T^n}k_t + \frac{T_v^b}{T_v^n}k_v + \frac{T_d^b}{T_d^n}k_d\right);$$

$$Q^s = \frac{1}{2}\left(\frac{T_s^b}{T_s^n}k_s + \frac{T_f^b}{T_f^n}k_f\right).$$

Здесь фигурируют отношения между новым и базовым вариантами разработок, где параметры: $Y^n, T^n, T_v^n, T_d^n, T_s^n, T_f^n$ – выход годной продукции, время создания изделия для его продажи на рынке, время верификации проекта, время диагностирования дефектных блоков и восстановления работоспособности, время исправного моделирования и моделирования дефектов; коэффициенты $k_y, k_t, k_v, k_d, k_s, k_f$ регламентируют вклад каждого компонента в общую эффективность предложенных разработок, где веса коэффициентов [0,1] определяются пользователем или экспертом в области проектирования.

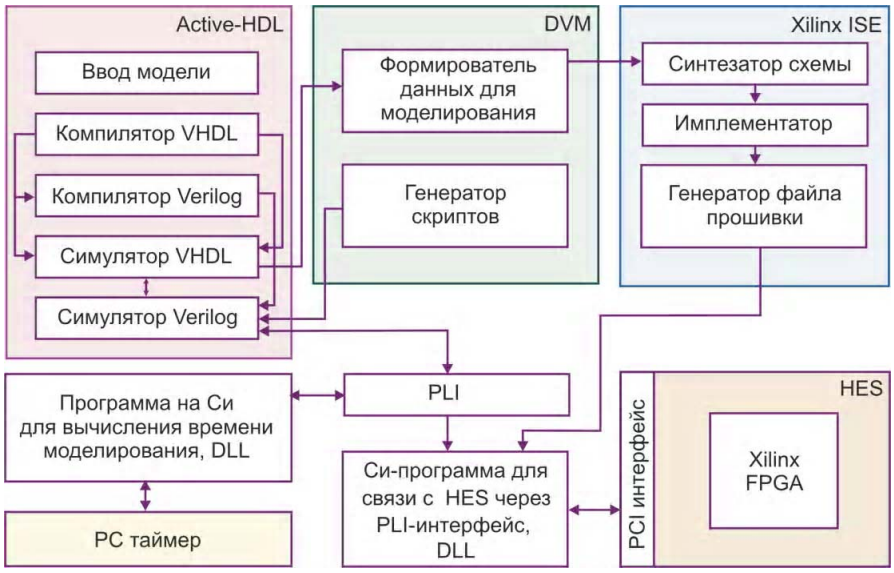

Рис. 4. Инфраструктура HW-ускоренного тестирования проекта

Оценка эффективности в соответствии с предложенными критериями, где все коэффициенты равны 1, проводилась на пяти реальных проектах, взятых из библиотек компаний Xilinx и Aldec. Информация, доступная для проведения статистических исследований, представлена в табл. 1.

Таблица 1. Эффективность инфраструктуры верификации и тестирования

Parameters / Projects	Code	Yield	TTM	VT	FFS	FS	DT	Q^t	Q^s
UART	1300	97/95	140/110	70/55	2500/35	5500/160	56/28	1,145	36
Wavelet–X	4300	94/91	220/160	120/100	4900/75	9900/230	76/33	1,115	37
JPEG 2000	5400	95/94	250/190	150/120	3300/45	6600/150	83/39	1,13	56,38
DSP-Aldec	11000	97/95	270/180	170/150	6500/95	12000/270	94/54	1,075	38,2
Golden Eye	13000	93/90	350/300	250/210	7500/140	15000/390	108/62	1,115	31,3

Здесь оцениваются параметры и характеристики (Yield – выход годной продукции (в %), Time-to-Market – время выхода изделия на рынок (в днях), Verification Time – длительность процесса верификации (в днях), Fault-Free Simulation – время исправного моделирования (в секундах), Fault Simulation – длительность процесса моделирования неисправностей (в секундах), Diagnosis Time – время диагностирования (в миллисекундах), Q^t – интегральная оценка эффективности, Q^s – интегральная оценка производительности инфраструктуры относительно длины HDL кода (Code), который непосредственно влияет на аппаратные затраты проектируемого цифрового изделия).

В соответствии со статистическими данными, представленными в табл. 1, ниже приведены графики: эффективности разработанной инфраструктуры (рис. 5), а также кривые наиболее существенных четырех параметров. Базовые варианты не имеют аппаратных ускорителей (Active HDL), разработанный прототип включает плату повышения производительности процесса моделирования компании Aldec.

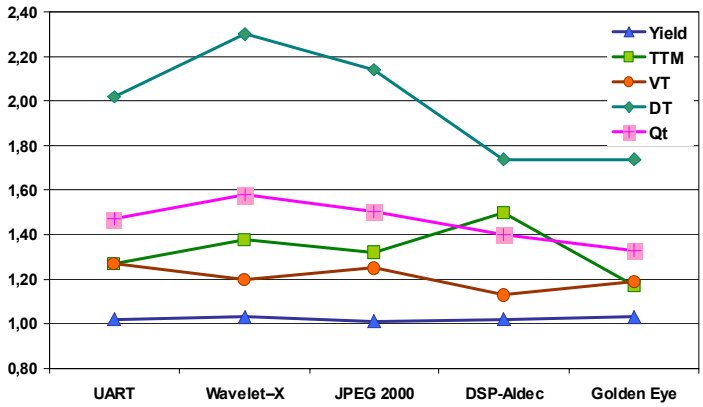

Рис. 5. Эффективность инфраструктуры верификации и тестирования

Повышение эффективности проектирования и эксплуатации цифрового изделия достигается за счет: 1) введения (5%) ассерционной избыточности в программный код функциональности; 2) дополнительных временных затрат – до 5%, взятых от времени создания системной HDL-модели, для построения транзакционного графа и выбора контрольных точек; 3) введения (5%) аппаратной избыточности в виде тестовой инфраструктуры на основе стандарта IEEE 1500 SECT для сервисного обслуживания функциональных модулей цифровой системы на кристалле в процессе ее эксплуатации.

Иерархию верификационной среды можно представить в виде организации компонентов (рис. 6).

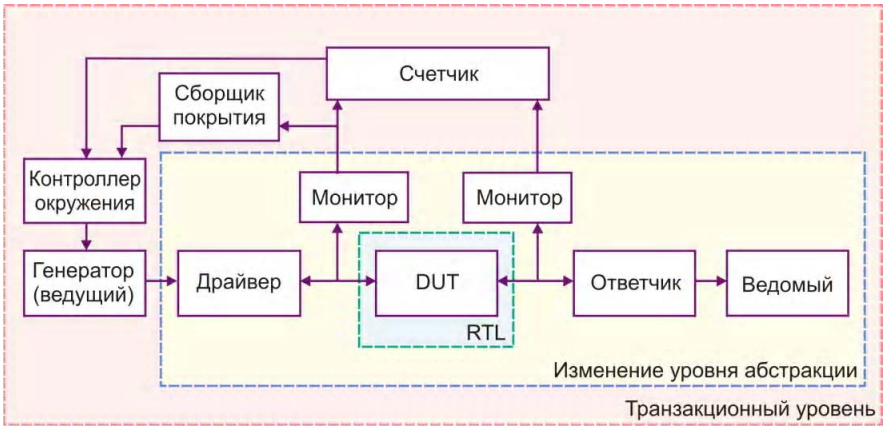

Рис. 6. Организация верификационного окружения

Внутренняя часть есть DUT компонент RTL-уровня абстракции, к которому подключается верификационное окружение TLM-уровня через верификационные компоненты-транзакторы. Роль каждого из них – конвертирование потока данных TLM-уровня для их восприятия на RTL-уровне, и наоборот. Окружение (см. рис. 6) – набор высокоуровневых

компонентов с транзакционными интерфейсами, которые предоставляют все необходимое для функционирования DUT. Компоненты окружения – генераторы тестов, ведущие (masters) и ведомые (slaves) блоки.

Генераторы тестовых воздействий создают поток транзакций для моделирования DUT. Они могут быть псевдослучайными, детерминированными или смешанными; самостоятельными или иметь управляющие сигналы; независимыми или синхронизированными. Простейший генератор выполняет псевдослучайную функцию синтеза тестов, которые передаются драйверу. Генератор сценариев создает детерминированные или смешанные последовательности, которые направлены на инициализацию специфических функций тестируемым устройством. Ведущий – двунаправленный компонент – отсылает запросы и принимает отклики, инициализирует активность, анализирует реакции для определения следующего сценария. Ведомый – двунаправленный компонент, обрабатывает запросы и возвращает отклики. Анализаторы (scoreboard – золотая модель, coverage collector – корзина тестового покрытия транзакций, адресного пространства, количества ошибок) получают информацию о верификационной среде и делают заключение о правильности и окончании процесса верификации. Контроллер формирует поток данных в верификационной среде и управляет ее активностью, пересылает данные от счетчика и коллектора покрытия компонентам окружения, запускает и останавливает генератор тестов.

3. Анализ тестопригодности для реальных HDL-проектов

Достаточно существенная избыточность HDL-модели предполагает ее эффективное использование в целях повышения тестопригодности структуры разработанного или написанного кода. Существующие стандарты тестопригодного проектирования аппаратуры (IEEE 1500, 11.49) можно адаптировать к верификации HDL-кода системных и регистровых программных моделей. Для этого используется модифицированный граф регистровых или транзакционных передач С.Г. Шаршунова [24-25], который предоставляет пользователю информацию о взаимосвязях булевых и регистровых переменных, памяти и интерфейсных шин, называемых транзакторами. Данные для синтеза графа получаются путем синтаксического анализа строк HDL-кода на предмет установления источников и приемников информации, которые являются вершинами (транзакторами) графа, соединенными ориентированными дугами. Каждая из дуг может быть отмечена количеством нагруженных на нее операторов. Построенный таким образом транзакционный граф (TG – Transaction Graph) [24-25] покрывает все функциональности (транзакции) программной модели и задает связи между вершинами, которые соответствуют приему, передаче и преобразованию информации.

Роль транзакционного графа заключается в создании модели передачи данных в целях определения тестопригодности всех вершин. Затем выделяется подмножество вершин, которое имеет минимальное значение тестопригодности, удовлетворяющее условию: $V = \forall i[Q(V_i) \le Q\min] \to Q^t \ge Q^t_{\min}$, где Q^t – качество покрытия тестом (testbench) неисправностей (корзины функциональных покрытий) при модификации структуры цифрового проекта или HDL-кода путем дополнительного мониторинга состояний критических вершин, для которых аппаратурная (программная) избыточность в реальных проектах составляет порядка 5% (Yervant Zorian).

Основными критериями эффективности проектирования изделия на рынке EDA являются выход годной продукции (Y – Yield) и относительное время создания продукта T^{Δ} – time-to-market. Совместно с относительными аппаратурными затратами проекта Z^{Δ} они формируют оценку E эффективности проектирования цифрового изделия, представленную в (1), как приведенное к общим затратам время, необходимое для создания программной и аппаратной функциональности $T^{\Delta}(S), T^{\Delta}(H)$, умноженное на аналогично приведенные программные и аппаратные затраты $Z^{\Delta}(S) \times Z^{\Delta}(H)$, а также на выход годной продукции Y. Последний параметр $Y = (1-P)^{n(1-Q)}$ зависит от тестопригодности (качества) проекта Q,

вероятности P существования в кристалле неисправных областей и числа необнаруженных дефектов n или D. Критерий временных затрат T^{Δ} также определяется тестопригодностью проекта Q и размерностью его программной Z(S), аппаратной модели Z(H), приведенной к дневной или часовой производительности $Z^1(S)\,[Z^1(H)]$ разработчика кода (аппаратуры). Коэффициенты k_q, k_w задают части временного интервала, необходимого для написания HDL-кода – $k_w = 0{,}3$ и верификации – $k_q = 0{,}7$ проекта, $k_q + k_w = 1$. Параметры $T^+(S), T^+(H)$ определяют время создания программной и аппаратной функциональностей, а $T^-(S), T^-(H)$ – затраты для их сервисного обслуживания, которые включают следующие компоненты:

$T(F^s), T(T^s), T(A), T(G^s)$ $[\,T(F^h), T(T^h), T(B), T(G^h)\,]$ – дополнительный период времени на создание функционального покрытия, тестовых последовательностей testbench, механизма ассерций (регистра граничного сканирования) и транзакционного графа программной (аппаратной) модели. Размерность HDL-кода и сложность аппаратуры для проекта определяется программными (аппаратными) компонентами:

$$Z^{\Sigma}(S) = Z(S), Z(F^s), Z(T^s), Z(A), Z(G^s)\ \ [\,Z^{\Sigma}(H) = Z(H), Z(F^h), Z(T^h), Z(B), Z(G^h)\,],$$

где в правой части равенства представлены компоненты: функциональность, функциональное покрытие, testbench (тест), механизм ассерций (регистр граничного сканирования), транзакционный граф программной (аппаратной) модели. Общая модель эффективности проектирования SoC имеет вид:

$$
\begin{aligned}
&E = \frac{1}{3}(Y + T^{\Delta} + Z^{\Delta}),\\
&Y = (1-P)^{n(1-Q)};\ D = 1 - Y^{(1-Q)};\\
&T^{\Delta} = T^{\Delta}(S) \times T^{\Delta}(H);\\
&Z^{\Delta} = Z^{\Delta}(S) \times Z^{\Delta}(H);\\
&T^{\Delta}(S) = \frac{T^+(S)}{T^+(S) + T^-(S)};\ T^{\Delta}(H) = \frac{T^+(H)}{T^+(H) + T^-(H)};\\
&T^+(S) = \left(k_w + k_q \frac{1-Q}{1+Q}\right)\frac{Z(S)}{Z^1(S)};\\
&T^-(S) = \frac{1-Q}{1+Q}\left[\frac{Z(F^s)}{Z^1(F^s)} + \frac{Z(T^s)}{Z^1(T^s)} + \frac{Z(A)}{Z^1(A)}\right] + \frac{Z(G^s)}{Z^1(G^s)};\\
&T^+(H) = \left(k_w + k_q \frac{1-Q}{1+Q}\right)\frac{Z(H)}{Z^1(H)};\\
&T^-(H) = \frac{1-Q}{1+Q}\left[\frac{Z(F^h)}{Z^1(F^h)} + \frac{Z(T^h)}{Z^1(T^h)} + \frac{Z(B)}{Z^1(B)}\right] + \frac{Z(G^h)}{Z^1(G^h)};\\
&T(S) = T^+(S) + T^-(S);\ T(H) = T^+(H) + T^-(H);\\
&Z^{\Delta}(S) = \frac{Z(S)}{Z(S) + Z(F^s) + Z(T^s) + Z(A) + Z(G^s)};\\
&Z^{\Delta}(H) = \frac{Z(H)}{Z(H) + Z(F^h) + Z(T^h) + Z(B) + Z(G^h)};
\end{aligned}
\qquad (1)
$$

$$Z^{\Sigma}(S) = Z(S) + Z(F^{s}) + Z(T^{s}) + Z(A) + Z(G^{s});$$

$$Z^{\Sigma}(H) = Z(H) + Z(F^{h}) + Z(T^{h}) + Z(B) + Z(G^{h});$$

$$Q = \{Q^{h}, Q^{s}\} = \frac{1}{n}\sum_{i=1}^{n}(U_i \times N_i);$$

$$U_i = \frac{1}{\tau}\sum_{j=1}^{x_i} T_j^i \times \frac{1}{d_i^x \vee t_i^x};\ N_i = \frac{1}{\tau}\sum_{j=1}^{y_i} T_j^i \times \frac{1}{d_i^y \vee t_i^y}.$$

В (1) интегральная оценка тестопригодности $Q = \{Q^{h}, Q^{s}\}$ транзакционных графов HDL-кода и аппаратной модели регистрового уровня функционально зависит от управляемости и наблюдаемости их вершин U_i, N_i, где n – количество вершин транзакционного графа. Управляемость и наблюдаемость есть метрика оценивания тестопригодности (культуры структуризации) не соединительных линий, а компонентов HDL-кода, таких как: регистр, счетчик, память или массивы, вход-выходные шины, векторы, логические или арифметические переменные цифрового проекта.

Управляемость вершины имеет функциональную зависимость от структурной глубины d_i^x нахождения транзактора относительно входов или длины конъюнктивного терма – t_i^x. Наблюдаемость имеет аналогичную зависимость $d_i^y \vee t_i^y$ относительно выходов. Для подсчета тестопригодности можно использовать один из параметров $d_i^x, t_i^x (d_i^y, t_i^y)$. Оценки управляемости и наблюдаемости зависят также от процентного отношения числа команд $\frac{1}{\tau}\sum_{j=1}^{x_i} T_j^i$, имеющих входной (выходной – $\frac{1}{\tau}\sum_{j=1}^{y_i} T_j^i$) доступ к вершине при анализе данной программы, к общему количеству команд τ, где x_i – число команд, формирующих доступ к входной вершине; (y_i) – количество команд, определяющих вершину как источник данных. Тестопригодность Q, представленная в (1), зависит от управляемости U, наблюдаемости (N), а также от стоимости реализации (Z) компонентов: метрики функционального покрытия (F), testbench (B), механизма ассерций (A), функциональности (S). Управляемость (наблюдаемость) есть функция от числа операторов, входящих в вершину (исходящих из вершины) транзакционного графа, а также от структурной глубины рассматриваемого элемента – расстояния от входов (выходов) или от количества временных тактов, необходимых для управления (наблюдения) компонента в заданном состоянии на временной оси. Влияние мощности L^m линий мониторинга проекта на изменение (увеличение, уменьшение) всех существенных параметров процесса проектирования цифровой системы на кристалле определяется следующим выражением:

$$[(L^m \uparrow \rightarrow (Z(B)\uparrow, Z(A)\uparrow, T(B)\uparrow, T(A)\uparrow)] \rightarrow [(Y\uparrow, Q\uparrow, D^f\uparrow)\,\&$$
$$\&\,(T^{\Delta}\downarrow, T^{sim}\downarrow, T^{t\_gen}\downarrow, T^{diag}\downarrow, T^{t\_hw}\downarrow, T^{v\_sw}\downarrow, L^{ud\_f}\downarrow)]. \qquad (2)$$

Вербальное и последовательное пояснение всех символов, входящих в выражение – увеличение числа точек наблюдения в программе или аппаратуре происходит за счет добавления аппаратной избыточности в виде регистра граничного сканирования, программной избыточности в виде механизма ассерций, а также за счет дополнительного времени создания упомянутых компонентов. Это дает возможность существенно увеличить выход годной продукции, тестопригодность проекта, глубину диагностирования дефектов и ошибок. Кроме того, существенно уменьшается время: выхода изделия на рынок, моделирования неисправностей, генерации тестов, диагностирования дефектов и ошибок, тестирования функциональности аппаратуры и верификации программного кода. Также существенно уменьшается число необнаруженных дорогостоящих дефектов и ошибок, которое влияет на выход годной продукции Yield.

Для численной оценки влияния параметров на качество и период проектирования, заданных в (2), ниже вводится интегральная нормированная средняя аддитивная оценка эффективности процесса проектирования. Она представлена суммой функционалов, формирую-

щих затратные (f_i^-) и выигрышные (f_i^+) функции, зависящие от числа линий мониторинга программного или аппаратного проектов:

$$E = \frac{1}{n}\sum_{i=1}^{n} k_i f_i,$$

$$f(L^m) = \begin{cases} \{f_1^- = Z(B);\ f_2^- = Z(A);\ f_3^- = T(B);\ f_4^- = T(A)\}; \\ \{f_5^+ = Y;\ f_6^+ = Q;\ f_7^+ = D^f\}; \\ \{f_8^+ = T^{\Delta};\ f_9^+ = T^{sim};\ f_{10}^+ = T^{t\_gen};\ f_{11}^+ = T^{diag}; \\ f_{12}^+ = T^{t\_hw};\ f_{13}^+ = T^{v\_sw};\ f_{14}^+ = L^{ud\_f}\}. \end{cases}$$

Здесь коэффициент k_i определяет весомозначность функционала в формировании интегральной оценки, которая используется для нахождения оптимизированной стратегии проектирования цифрового изделия.

Интегральной оценке E ставятся в соответствие графики зависимостей наиболее существенных функционалов от числа линий мониторинга, которые представлены на рис. 7.

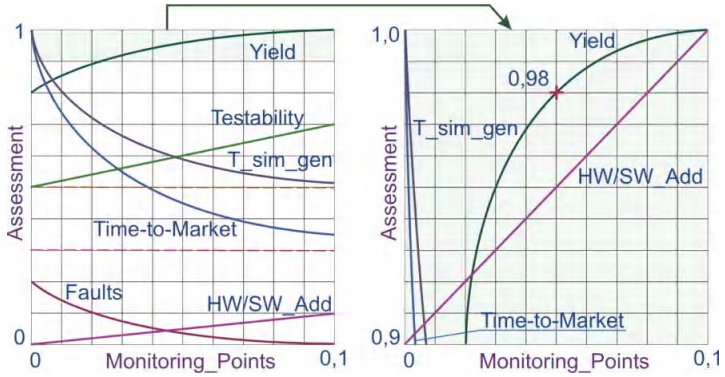

Рис. 7. Влияние точек контроля на эффективность проектирования

В левой части рис. 7 представлены зависимости эффективностей (Yield – выход годной продукции, Testability – тестопригодность проекта, T_sim_gen – время моделирования, генерации тестов и поиска дефектов, Time-to-Market – время выхода изделия на рынок, HW/SW_Add – дополнительные аппаратные и программные затраты для реализации IEEE 1500 регистра граничного сканирования и механизма ассерций, Faults – количество непроверенных дефектов) от числа линий наблюдения в интервале [0-0,1]. Правая часть рис. 7 иллюстрирует поведение отдельных функционалов в укрупненном масштабе (10:1) интервала [0,9-1] по оси ординат левого графика.

Для расчета суммарного эффекта от имплементации в проект дополнительных линий наблюдения и затрат (аппаратных и программных) необходимо модифицировать приведенный в (1) критерий эффективности к следующему виду:

$$E = \frac{1}{2 \times Z^{\Delta}}(Y^{\Delta} + T^{\Delta}).$$

Данный критерий определяет оценку практической значимости полезных приращений, приведенных к проценту аппаратурных и программных затрат, или эффект от разработанных моделей и методов тестопригодного проектирования, моделирования, тестирования и верификации на 5 реальных проектах. Все существенные параметры сведены в табл. 2.

Средняя оценка эффекта для пяти проектов равна 7,83. Это означает, каждый процент аппаратурной, программной избыточности приносит почти восемь процентов эффекта по времени моделирования, тестирования, верификации и качеству изделия. Графики приращения данных параметров проектирования в функциональной зависимости от избыточности представлены на рис. 8.

4. Выводы

Научная новизна и практическая значимость.

1. С учетом результатов, опубликованных ранее в [3, 16-25], достигнута основная цель исследования, которая заключается в существенном уменьшении времени верификации HDL-моделей и тестирования аппаратных компонентов SoC путем создания инфраструктуры анализа проекта, позволяющей: 1) оценивать тестопригодность программно-аппаратных модулей путем построения транзакционного графа для использования механизма ассерций и IEEE 1500 SECT стандарта; 2) модифицировать структуру проекта путем введения дополнительных точек мониторинга; повышать производительность сервисных средств моделирования, диагностирования и восстановления работоспособности программных и аппаратных компонентов цифровых систем на кристаллах.

Таблица 2. Приращения параметров проектирования

Design	Code	Ntk	%tk	ΔQH/QS	ΔAddH/S	ΔTsim	ΔTgen	ΔYield	ΔTTM	E
UART	1300	52	0,04	0,08	0,047	0,375	0,275	0,145	0,505	6,9
Wavelet–X	4300	167	0,04	0,085	0,045	0,365	0,265	0,14	0,5	7,1
JPEG 2000	5400	189	0,04	0,075	0,044	0,34	0,234	0,135	0,485	7,05
DSP-Aldec	11000	374	0,03	0,07	0,035	0,35	0,235	0,13	0,475	8,6
Golden Eye	13000	390	0,03	0,06	0,03	0,33	0,233	0,12	0,45	9,5

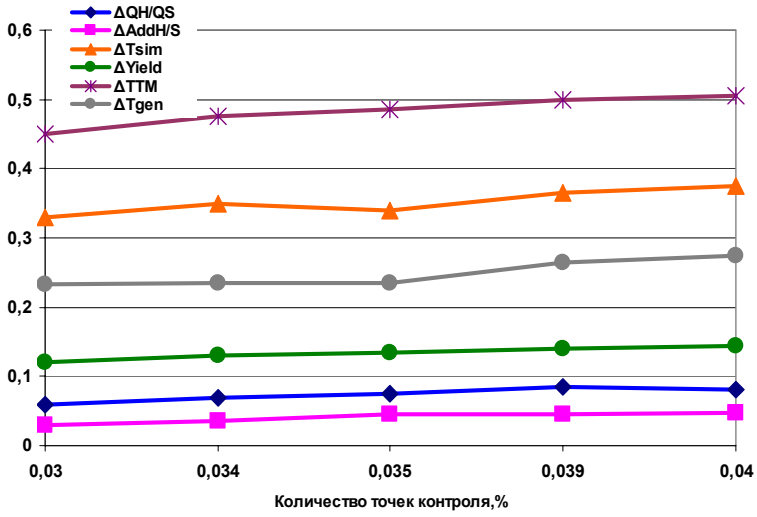

Рис. 8. Графики приращения параметров проектирования

2. Предложена инфраструктура верификации цифровых систем на кристаллах, которая является инвариантной (универсальной) по отношению к прототипу и HDL-модели и содержит следующие компоненты: 1) Transaction Graph Design for Testability – транзакционная модель программного кода. Здесь вершинами являются логические и регистровые переменные, векторы, массивы и память, а дугами – операторы кода, выполняющие транзакции между вершинами. Транзакционный граф в части верификации прототипа строится и для аппаратных компонентов системы. 2) Testability Assessment for Transaction Graph Nodes – вычислительные модели процессов оценки тестопригодности вершин транзакционного графа HDL-кода и прототипа. Они необходимы для выполнения процедуры выбора достаточного количества вершин транзакционного графа с минимальной оценкой тестопригодности, в которые устанавливаются ассерции (для HDL-модели) путем использования стандарта IEEE 1500 SECT для прототипа.

3. Инфраструктура верификации интегрирована с программой моделирования QuestaSim для исправления ошибок в процессе создания HDL-модели. Проведена валидация работоспособности инфраструктуры отладки программного кода совместно с системой моделирования Active HDL, Aldec, использующей аппаратный акселератор. Осуществлена оценка эффективности инфраструктуры тестирования и верификации путем выполнения экспериментов над реальными цифровыми системами.

4. Средняя оценка суммарного эффекта от внедрения инфраструктуры встроенного тестирования для пяти реальных проектов равна 7,83, что означает: каждый процент аппаратурной, программной избыточности приносит почти восемь процентов суммарного эффекта, включающего время моделирования, тестирования, верификации, выхода продукции на рынок. Дальнейшие исследования необходимо связывать с объектно-ориентированной верификацией (ООВ), которая представляет собой актуальнейшую и динамически развивающуюся область. Ведущие компании EDA-индустрии предлагают свои продукты, позволяющие внедрить ключевые идеи ООВ в верификационное окружение. Самые стабильные и конкурентоспособные методологии разрабатываются такими вендорами как Synopsys (VMM) и Mentor Graphics (AVM), которые являются инициаторами появления и внедрения SystemVerilog. Несмотря на достаточно существенные отличия в подходах, VMM и AVM технологии верификации становятся широко применимыми стандартами дефакто. Таким образом, основной тренд развития современных структур отладки связывается с созданием объектно-ориентированных библиотек тестовых решений и разработкой конкурентоспособных методов верификации на высоком уровне абстракции.

Список литературы: 1. *Aldec*, Inc. Official web-site: http://www.aldec.com. **2.** Zorian Yervant. What is Infrustructure IP? // IEEE Design & Test of Computers. 2002. P. 5-7. **3.** *Хаханов В.И.* Проектирование и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. Харьков: ХНУРЭ, 2009. 484с. **4.** *Bergeron Janick.* Writing testbenches: functional verification of HDL models. Boston: Kluwer Academic Publishers. 2001. **5.** *IEEE* 1500 Web Site. http://grouper.ieee.org/groups/1500. **6.** *Harry Foster, Adam Krolnik, David Lacey.* Assertion-based design. Kluwer Academic Publishers. Springer. 2005. 392 p. **7.** *Tabatabaei Sassan, Ivanov Andre.* Embedded Timing Analysis: A SoC Infrastructure // IEEE Design and Test of Computers. 2002. P. 24-36. **8.** *Abramovici M., Breuer M.A. and Friedman A.D.* Digital System Testing and Testable Design.– Computer Science Press. 1998. 652 p. **9.** *Marinissen E. J., Zorian Y.* IEEE Std 1500 Enables Modular SoC Testing // Design & Test of Computers.Jan./Feb. 2009. P.8-16. **10.** *Shoukourian S., Vardanian V., Zorian Y.* SoC Yield Optimization via an Embedded-Memory Test and Repair Infrastructure // IEEE Design and Test of Computers. 2004. P. 200-207. **11.** *IEEE* standard 1076-2000, "IEEE Standard VHDL Language Reference Manual", January 2000. **12.** *IEEE* Std 1800-2005. IEEE Standard for System Verilog. **13.** *Synopsys*, Inc. Synopsys Delivers First Complete SystemVerilog Design and Verification Flow. Official web site, 2006. **14.** *Mentor* Graphics, Inc. Mentor Graphics Announces HDL Designer Series with SystemVerilog Support for Design-to-Verification Productivity. Official web site, 2007. **15.** *Janick Bergeron, Eduard Cerny, Alan Hunter, Andrew Nightingale.* Verification Methodology. Manual for SystemVerilog. Springer. 2005. **16.** *Шаршунов С.Г.* Построение тестов микропроцессоров. 1. Общая модель. Проверка обработки данных // Автоматика и телемеханика. 1985. №11. С. 145-155. **17.** *Hahanov V.* General testing models of SoC hardware-software components // Radioelectronics & Informatics. 2008. №1. P. 88-95. **18.** *Хаханов В.И.* Восстановление работоспособности встроенной памяти SoC // Науково-технічний журнал «Інформаційно-керуючі системи на залізничному транспорті». 2008. № 4. P. 120-127. **19.** *Hahanov V.* Embedded Method of SoC Diagnosis / V. Hahanov, E. Litvinova, V. Obrizan, W. Gharibi // Electronics and Electrical Engineering. 2008. № 8(88). P. 3-8. **20.** *Hahanov Vladimir.* Algebra-Logical Diagnosis Model for SoC F-IP / Vladimir Hahanov, Vladimir Obrizan, Eugenia Litvinova, Ka Lok Man // WSEAS Transactions on Circuits and Systems.– Issue 7, Vol. 7. July, 2008. P. 708-717. **21.** *Хаханов В.И.* Сервисное обслуживание современных цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, Ngene Christopher Umerah // Радиоэлектронные и компьютерные системы. 2009. № 7 (41). С. 319-323. **22.** *Литвинова Е.И.* Технологии диагностирования и восстановления System-in-Package / Е. И. Литвинова // АСУ и приборы автоматики. 2009. № 146. С. 4–21. **23.** *Литвинова Е.И.* Метод покрытия неисправных логических блоков цифровых систем на кристаллах ремонтными клетками // Радиоэлектроника и информатика. 2009. №1. С. 46-49. **24.** *Хаханов В.И.* Технология покрытия дефектных блоков резервными компонентами / В.И. Хаханов, С.В. Чумаченко, Е.И. Литвинова, О.В. Захарченко// АСУ и приборы автоматики. 2009. Вып. 147. С. 52-64. **25.** *Хаханов В.И.* Тестирование и верификация HDL-моделей компонентов SOC. I. / В.И. Хаханов, Е.И. Литвинова, С.В. Чумаченко, И.А. Побеженко, Ngene Christopher Umerah // Радиоэлектроника и информатика. 2009. № 3. С. 45-52. **26.** *Хаханов В.И.* Тестирование и верификация HDL-моделей компонентов SOC. II. / В.И. Хаханов, Е.И. Литвинова, И.А. Побеженко, Tiecoura Yves // АСУ и приборы автоматики. 2009. Вып. 148. С. 26-37.

Поступила в редколлегию 28.08.2009

Литвинова Евгения Ивановна, канд. техн. наук, доцент кафедры ТАПР ХНУРЭ. Научные интересы: алгоритмизация задач автоматизированного проектирования электронных вычислительных средств, автоматизация диагностирования и встроенный ремонт компонентов цифровых систем в пакете (SiP). Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-421, e-mail: kiu@kture.kharkov.ua.