УДК519.713:681.326

ИЕРАРХИЧЕСКИЙ МЕТОД ТЕСТИРОВАНИЯ ПРОГРАММНЫХ ПРОДУКТОВ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ АССЕРЦИОННЫХ БИБЛИОТЕК

ХАХАНОВ В.И., КАМИНСКАЯ М.А., СУШАНОВ А.В.

Предлагается метод анализа тестопригодности программных продуктов, представленных в виде композиции операционного и управляющего автоматов, а также методология выбора контрольных точек для дальнейшего внедрения ассерций в программный код устройства.

1. Введение

Современные цифровые системы на кристалле – это сложные системы, состоящие из большого количества взаимодействующих компонентов. Структурная и функциональная сложность таких систем, трудоемкость процедуры тестирования повышает вероятность появления ошибки при разработке устройства, поэтому верификация цифровых устройств становится все более актуальной темой. Актуальность определяется необходимостью повышения быстродействия средств моделирования, улучшения качества теста и уменьшения его размерности для цифровых систем на кристаллах, имеющих миллионы вентилей. Высокие затраты, обусловленные трудоемкостью верификации функционально и структурно сложных схем, могут достигать 70% от общего времени разработки проекта.

Поэтому для обеспечения надежного функционирования систем на кристалле необходимо проводить тестирование и диагностирование неисправностей на самых ранних этапах разработки цифрового устройства с минимальными временными затратами на тестирование.

Любую сложную цифровую систему можно рассматривать как систему, имеющую несколько уровней иерархии. На каждом уровне разработки должна быть проведена процедура верификации и тестирования. В технологиях тестирования сложных цифровых устройств могут быть использованы такие подходы как: Design for Manufacturability, Design for Testability, Design for Verification [1-15].

Здесь необходимо дать определение тестопригодности устройства. Тестопригодность – свойство изделия, направленное на соблюдение в установленных пределах материальных и временных затрат показателей, характеризующих приспособленность объекта к диагностированию, генерации тестов, моделированию дефектов за счет увеличения оборудования и дополнительных временных затрат на отдельных этапах проектирования. Методы проектирования тестопригодных схем делятся на:

1) анализ структурно-функционального исполнения объекта, численная оценка управляемости и наблюдаемости как меры тестопригодности схемы, которая может использоваться на этапе проектирования;

2) способы структурного проектирования тестопригодных схем, в пределе самотестируемых, основанных на использовании свойств пути сканирования, обеспечивающего доступ к внутренним точкам схемы.

Важным этапом процесса разработки цифровых устройств является процедура верификации, которая необходима для анализа и устранения всех ошибок как можно на более ранней стадии, что приводит к значительному уменьшению временных и материальных затрат (рис. 1).

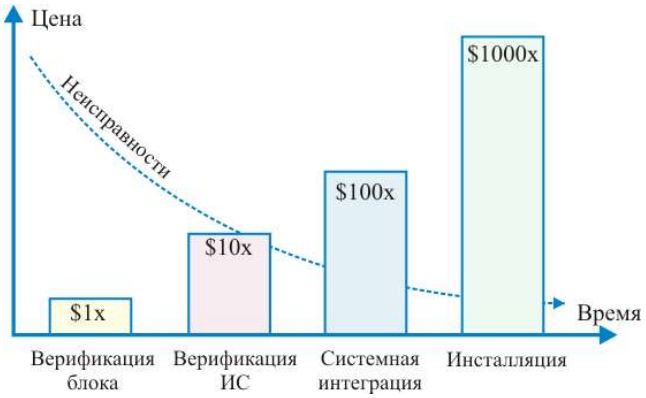


Рис. 1. Зависимость стоимости процедуры верификации от уровня разработки

Использование подходов тестопригодного проектирования на самой ранней стадии разработки (системный уровень) позволяет существенно повысить качество выпускаемого продукта и эффективность процедуры тестирования за счет внесения допустимой аппаратурной и временной избыточности. Временная избыточность определяется использованием ассерций в программном коде, что вызывает незначительную задержку при проектировании кода, однако такой подход позволит существенно уменьшить временные затраты на этапе тестирования и ускорить процесс выхода устройства на рынок (time to market).

При проектировании на системном уровне используются языки описания аппаратуры (hardware description language – HDL) для представления функционального поведения проекта без идентификации структуры ло-

гических элементов и связей между ними. К HDL языкам относятся VHDL, System C и Verilog. К достоинствам проектирования на языках описания аппаратуры можно отнести повторное использование готовых проектов или их частей.

При проведениии процедуры верификации и тестирования самый адекватный анализ соответствует наиболее точной модели, которая определяется вентильным уровнем описания, поскольку структура устройства здесь представлена максимально детализированно. Тем не менее, анализ неисправностей на более высоких уровнях описания, где модель проекта отражает лишь структуру взаимосвязанных компонентов, имеет место быть, поскольку здесь трудоемкость процедуры анализа минимальна, но использование ассерций и последующая модификация проекта на основе технологий граничного сканирования могут существенно повлиять на стоимость диагностического обеспечения и обслуживания (временные и материальные затраты на синтез тестов, моделирование неисправностей и диагностирование дефектов для каждой стадии проектирования).

Каждый уровень иерархии имеет название и базовый набор структурных примитивов.

На рис. 2 представлена структура цифрового устройства, где видны уровни описания модели устройства и технологии, которые могут быть применены для каждого уровня представления.

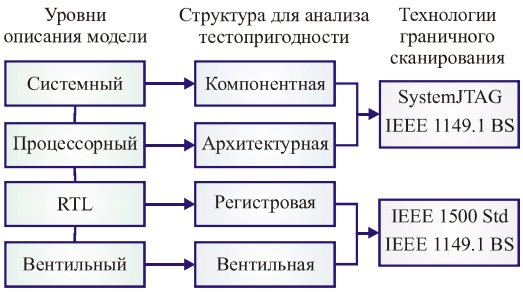


Рис. 2. Уровни представления цифровых устройств

Для вентильного уровня – это логические элементы (And, Or, Not, Xor). Для регистрового уровня таковыми являются: регистры, счётчики, мультиплексоры, АЛУ, дешифраторы. На системном уровне устройство представлено в виде совокупности взаимосвязанных компонентов (функциональных блоков) или операционного и управляющего автоматов (граф-схема алгоритма). Далее рассматриваются модели проекта системного уровня, представленные в виде граф-схем. Здесь основная сложность их тестирования заключается в том, что в графе должны быть проверены все блоки, которые возможно были неправильно описаны и вследствие чего неправильно функционируют (например, замена одного оператора другим), трудно достижимые состояния; тупиковые ситуации или коллизии двух потоков данных; места локализации ветвлений и обратных связей в коде (if, case, loop).

По принципу академика В.М. Глушкова любое устройство обработки цифровой информации можно разделить на операционный и управляющий блоки. Такой подход упрощает проектирование, а также облегчает понимание процесса функционирования вычислительного устройства.

Операционный автомат может быть представлен системным уровнем (досинтезное представление устройства, его внутренняя структурная модель), уровнем регистровых передач, а также вентильным уровнем. Для проведения анализа тестопригодности операционный автомат может быть представлен в виде графа, содержащего вершины и направленные дуги, где каждая вершина определена на множестве компонентов: входные переменные, выходные переменные, регистровые переменные, блок АЛУ, массивы памяти, представленные в формате их задания в программе. Дуги представляют собой совокупность операндов (или операций), которые осуществляют перенос информации между вершинами с преобразованием или без него.

На рис. 3 представлена структура цифрового устройства в виде совокупности управляющего и операционного автомата.

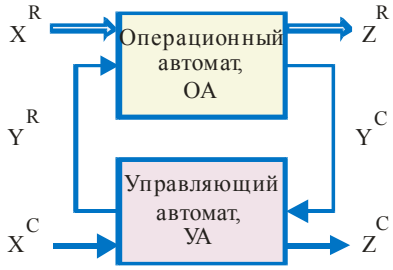


Рис. 3. Уровни представления цифровых устройств

Предлагаемая автоматная модель представлена двумя основными компонентами M^{OA}, M^{YA} – операционный и управляющий автомат. Данному рисунку ставится в соответствие следующая аналитическая абстрактная структура:

$$\begin{cases} M = (M^{OA}, M^{CA}); \\ M^{OA} = \{\vec{X}^{O}, Y^{C}, Y^{O}, \vec{Z}^{O}\}; \; M^{CA} = \{X^{C}, Y^{O}, Y^{C}, Z^{C}\}; \\ \vec{Z}^{O} = f^{O}(\vec{X}^{O}, Y^{C}); Y^{O} = g^{O}(\vec{X}^{O}, Y^{C}); \\ Z^{C} = f^{C}(X^{C}, Y^{O}); Y^{C} = g^{C}(X^{C}, Y^{O}). \end{cases}$$

Здесь $\vec{X}^{O}, \vec{Z}^{O}$ – векторы или регистровые входные и выходные переменные; Y^{C}, Y^{O}, Z^{C} – сигналы управления (инициализации) операций, оповестительные сигналы, сигналы мониторинга управляющего автомата соответственно; $f^{O}, g^{O}(f^{C}, g^{C})$ – функции, определяющие отношения между интерфейсными сигналами в операционном и управляющем автоматах.

В работе предлагается структурный метод анализа цифрового устройства или программного продукта, представленного в виде программного кода на системном уровне, который позволит существенно уменьшить временные затраты на процедуру тестирования и верификации и увеличить процент выхода годных к эксплуатации устройств за счет введения дополнительных ассерций в код устройства.

Объектом исследования является цифровое устройство или программное обеспечение, представленное в виде программного кода на системном уровне.

Задачи: 1. Построение автоматной модели программного продукта или модуля, включающего операционный и управляющий автомат. 2. Синтез ГСА, управляющего вычислительными процессами и передачей данных программного модуля. 3. Синтез графа операционного автомата С.Г. Шаршунова [16,17], который определяет вычислительные процессы и потоки данных. 4. Полная модель, представленная графом регистровых передач, охватывает все операторы программного модуля, что необходимо для синтеза тестопригодного программного продукта в части передачи данных. 5. Разработка алгоритма модификации устройства для возможности повышения тестопригодности устройства (уменьшения количества непроверяемых неисправностей заданным тестом) за счет вычисления показателей тестопригодности и введения ассерций в программный код. 6. Апробация предложенного метода на примерах реальных устройств.

2. Развитие методов проектирования цифровых устройств

При переходе на более высокий уровень описания расширяются возможности проектировщика, увеличивается продуктивность процедуры тестирования и улучшается процесс передачи информации между соединенными блоками устройства. Важным моментом здесь является развитие и разработка новых технологий на основе спецификаций, а затем и разработка новых стандартов, которые открывают новые, более быстрые пути поступления изделия на мировой рынок. Так, разработка языков регистровых передач повлекла за собой возникновение средств синтеза цифровых устройств. Обеспечение корректности функционирования устройства на уровне регистровых передач является одной из самых больших трудностей для изготовителей современных систем на кристалле и заказных интегральных схем (ASIC). Главной целью разработчиков является сокращение времени верификации устройства, что требует разработки новых технологий проектирования и верификации. Различают динамическую верификацию (моделирование неисправностей) и статическую (формальную), которая включает в себя, к примеру, использование ассерций [18-20]. При развитии средств статической верификации автоматически становятся очевидными многие свойства проекта при структурном анализе его модели, представленной на уровне регистровых передач. Такие средства направлены на исчерпывающую верификацию свойств модели с помощью формальных техник. При умелом использовании технологий верификации существует возможность исправления практически всех неточностей проекта на самой ранней стадии его разработки.

Кроме несинтезируемых ассерций, которые выбираются на основе спецификации, существуют также различные Ad-Hoc технологии, т.е., введение в устройство дополнительной синтезируемой логики, не влияющей на работу устройства в режиме нормального функционирования, но обеспечивающей наблюдаемость внутренних «проблемных» частей (блоков) устройства или межсоединений (узких мест) в режиме сканирования. Такой подход хотя и требует дополнительных аппаратурных затрат, но позволяет существенно повысить процент покрытия неисправностей, а значит и повысить качество проектируемого устройства и ускорить его выход на рынок.

Чтобы понять необходимость введения и предназначение дополнительной сканирующей логики в устройство, необходимо ввести понятия наблюдаемости, управляемости и тестопригодности.

Наблюдаемость – способность наблюдать снимаемые с выходов устройства реакции на входные воздействия (тесты), которые генерируются с помощью тестовой программы testbench, т.е. возможность наблюдать результат воздействия на входные (и внутренние) линии кода или структуры. В этом случае testbench можно рассматривать как средство для “ограниченной” наблюдаемости, т.е. наблюдения только внешних портов устройства или модели.

Управляемость – возможность стимулировать (активизировать) определенную линию кода или структуры внутри проекта. В рамках управляемости можно рассматривать такие параметры как: покрытие линий кода, покрытие разветвлений кода, покрытие путей, покрытие выражений.

Покрытие линий кода – временной отрезок, за который определенная линия кода была выполнена (или нет) во время моделирования. *Покрытие разветвлений кода* – время разработки разветвлений кода (условные операторы, операторы IF, CASE и т.д.).

Покрытие путей – время обработки уникальных путей в коде (включая разветвления) во время моделирования.

Покрытие выражений – это показатель управляемости каждой отдельной переменной, которая связана со значением управляемости на выходной линии устройства.

В работе рассматривается понятие управляемости и наблюдаемости внешних и внутренних узлов преобразованного во внутреннюю модель программного кода и на основе полученных показателей предполагается введение в устройство тестирующей логики для труднодостижимых узлов. Выбранные контрольные точки с наихудшими показателями управля-

емости и наблюдаемости должны указывать на места в схеме, которые требуют дополнительной проверки. Такую процедуру может проводить любой человек, даже не участвующий изначально в проекте, поскольку анализ тестопригодности и выбор контрольных точек производится на внутренней модели устройства, в которую переводится код. В этом случае инженеру или разработчику необходимо только внести в разрабатываемый код дополнительную логику (в места, выбранные по предложенному методу). В случае же использования ассерций нет четкого алгоритма их применения, только разработчик спецификации и кода может определить, в какой именно части кода необходима ассерция. Новому программисту, который может сменить своего предшественника, придется заново разбираться в коде и самому «прочувствовать», где именно должна быть внедрена ассерция. Именно это и послужило поводом для разработки метода структурного анализа цифрового устройства и целенаправленного выбора узких мест в нем для введения контрольных точек (ассерций в программный код устройства).

В настоящее время ассерции завоевали широкую популярность благодаря возможности их использования практически в любом проекте и простоте их реализации. Ассерции могут применяться при разработке систем на кристалле, в разработке мобильных устройств, для разработки устройств, используемых в медицинских учреждениях.

В [21] описываются преимущества использования ассерций при автоматизированном тестировании, в частности описывается методология применения ассерций для анализа исключений, приводятся принципы правильного написания кода с использованием ассерций.

В [22] представлена методология использования ассерций для анализа производительности устройств мобильных телефонов. В работе особое внимание уделяется вопросу увеличения производительности таких устройств.

В работах [23,24] описывается методология использования System Verilog ассерций. Методология применения ассерций исчерпывающе представлена в [19] – OpenVera Assertions (OVA), PSL/Sugar, Open Verification Libraries (OVL) и C/C++/SystemC™ конструкции как в программном моделировании, так и аппаратно в виде синтезируемых конструкций. Для описания устройства могут служить такие языки как VHDL, System Verilog, C. Существует ряд стандартов использования ассерций, таких как стандарты AVM [18,20], VVM.

Ассерции моделируются отдельно от остального HDL кода. Результат работы ассерций можно увидеть в программе просмотра функционального покрытия (Functional Coverage Viewer).

Ассерция – есть высказывание системного уровня, определяющее корректность преобразований в процессе проектирования относительно входного описания текущего этапа или требований спецификации.

Следуя приведенному определению, можно записать ассерцию в виде предиката

$$L_i = f(L_{i1}, L_{i2}, ..., L_{ij}, ..., L_{in}) = Y = \{0,1\},$$

который на множестве n_i переменных принимает значения «истинно» или «ложно» [25].

Ассерции могут быть разделены на три типа: проверки неисправностей, функциональных путей и возможных состояний проектируемого изделия. Такое разделение позволяет проверить не только механизм вычислений (операционный автомат ЦУ), но и механизм управления (управляющий автомат), т.е. граф переходов, для которого необходимо проверять достижимость всех состояний и переходов [25–28]. Здесь можно говорить об анализе тестопригодности для графа управляющего автомата. В работе также предлагается механизм тестового диагностирования и алгоритм внедрения ассерций в код для операционного автомата, представленного в виде графа переходов.

3. Метод структурного анализа операционного автомата

Прототипом предлагаемой методологии послужило использование стандартов тестопригодного проектирования, таких как IEEE 1149.1 BS и IEEE 1500 SECT. На рис. 4 показана стратегия тестирования цифровых систем на кристалле, представленных аппаратной реализацией.

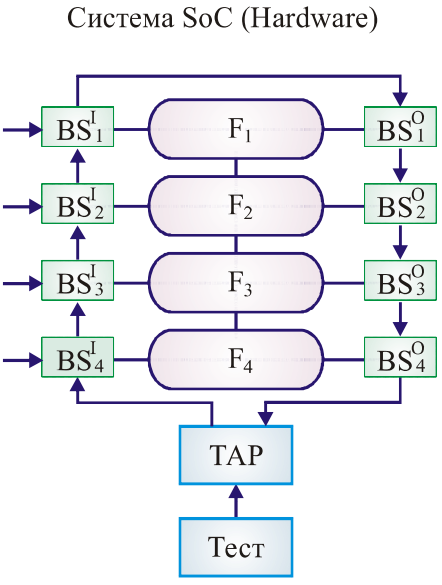


Рис. 4. Модель использования граничного сканирования для аппаратной реализации систм на кристалле

Здесь к каждому аппаратному функциональному блоку присоединяется ячейка сканирования. Загрузка тестовых данных может осуществляться как параллельно, так и последовательно, под управлением тестововго контроллера Test Access Port. В случае аппаратной реализации в качестве тестовых данных используются последовательности нулей и единиц. Та-

кой подход может быть успешно использован на уровне регистровых передач и вентильном уровне. Для вентильного уровня предполагается применение модифицированной ячейки сканирования [29]. Однако с ростом степени интеграции элементов на кристалле SoC все актуальнее становится использование ассерций на системном уровне проектирования, когда система представляет собой программный код.

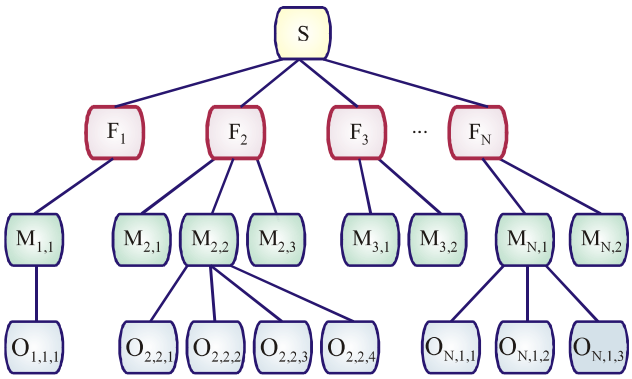


Рис. 5. Иерархическое представление SoC или программного продукта

Детализация SoC на системном уровне может быть представлена как совокупность функциональностей, каждая из них, в свою очередь, может быть разложена на совокупность более мелких функциональностей. Такое разделение показано на рисунке 5.

Первый, глобальный уровень, представляет собой отдельная система S (SoC либо любой проект, представленный на языках описания аппаратуры или языках программирования). На следующем уровне система S может быть представлена набором взаимосвязанных функциональных блоков F_i – в этом случае для каждого блока устанавливается ассерция $A(F_i)$ и системой диагностирования определяется, в каком именно функциональном блоке существует неисправность. Таким образом, может быть получен выходной вектор неисправностей, состоящий из нулей и единиц, где наличие нуля указывает на исправный функциональный блок $F_i(C)$, а наличие единицы – на неисправный функциональный блок $F_i(D)$. Далее рассмотрим неисправный блок системы. Каждый функциональный блок представляет собой метод либо набор методов $M_{i,j}$, что может быть представлено на следующем уровне иерархии. Каждому методу тоже могут соответствовать ассерции. Метод в свою очередь представляет собой набор операторов $O_{i,j,k}$. Такой подход может проводиться на этапе разработки, где отдельные функциональности покрываются unit тестами и позволяют минимизировать объем процедуры тестирования, проводимого тестировщиками, поскольку дополнительно тестируется уже не вся система, а ее отдельные блоки, в которых возникла неисправность.

На рис. 6 представлена архитектура диагностирования устройства на основе использования ассерций.

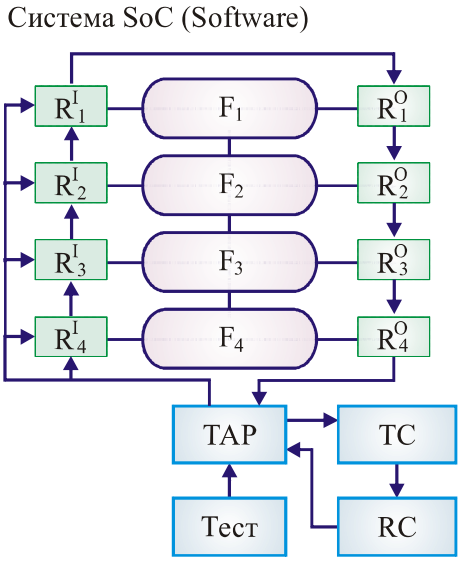


Рис. 6. Диагностирование SoC с использованием ассерций

Здесь F1-F4 – функциональности устройства, входные регистры RI используются для временного хранения тестовых данных, поступающих к функциональностям с помощью контроллера диагностирования. В выходных регистрах RO содержатся результаты тестирования – нули и единицы, единицы сигнализируют о том, что произошла ошибка в одной из функциональностей.

Выходные данные из регистра могут быть записаны в виде результирующего n-разрядного тест-вектора $V = \{V_1, V_2, \ldots, V_i, \ldots, V_n\}$, где n – количество функциональных блоков в устройстве.

Для определения минимального количества тестовых сценариев, необходимых для покрытия неисправного функционала, может быть использован алгебрологический метод, описание которого приведено ниже [30]. Таким образом, решение задачи диагностирования может быть получено путем применения булевой алгебры и таблицы неисправностей M, представляющей собой декартово произведение теста T на множество заданных дефектов F, в совокупности с вектором экспериментальной проверки V, где решение задачи покрытия дает максимально точный результат в виде ДНФ, а каждый терм есть возможный вариант наличия в устройстве дефектов.

Модель процесса диагностирования представлена компонентами:

$$A = < T, F, M, V >,$$

$$T = (T_1, T_2, \ldots, T_i, \ldots, T_n); \quad F = (F_1, F_2, \ldots, F_j, \ldots, F_m); \quad (1)$$

$$M = \left|M_{ij}\right|, i = \overline{1,n}; j = \overline{1,m}; \quad V = (V_1, V_2, \ldots, V_i, \ldots, V_n);$$

Значение координат вектора V есть результат выполнения операции тестирования и сравнения модельной и фактической реакций выходов устройства.

Решение задачи диагностирования сводится к анализу таблицы неисправностей, полученной в результате

моделирования дефектов, путем последующей записи логического произведения дизъюнкций (КНФ), записанных по единичным значениям строк таблицы неисправностей:

$$F = \bigwedge_{\forall V_i=1}^{i=\overline{1,n}} \left(\bigvee_{\forall M_{ij}=1}^{j=\overline{1,m}} F_j \right). \qquad (2)$$

Конъюнктивная нормальная форма (КНФ), полученная из ТН, трансформируется к дизъюнктивной нормальной форме (ДНФ) с помощью эквивалентных преобразований [31].

В результате получается булева функция, где термы – логические произведения представляют полное множество решений в виде сочетания дефектов (дающих по выходам SoC или ее компоненту – вектор экспериментальной проверки V):

$$F = \bigwedge_{\forall V_i=1}^{i=\overline{1,n}} \left(\bigvee_{\forall M_{ij}=1}^{j=\overline{1,m}} F_j \right) = \left| \begin{bmatrix} a \vee ab = b \\ a \vee a = a \end{bmatrix} \right| = \bigvee_{i=1}^{2^m} \left(\bigwedge_{j=1}^{m} k_j F_j \right), k_j = \{0,1\}. \qquad (3)$$

Функция (3) в общем случае формирует диагноз в виде некоторого подмножества сочетаний (кратных) дефектов, которые далее нуждаются в уточнении путем применения дополнительного зондирования внутренних точек с помощью регистра граничного сканирования. Количество единиц в векторе экспериментальной проверки V формирует число дизъюнктивных термов КНФ (3). Каждый терм – построчная запись дефектов (через логическую операцию ИЛИ), оказывающих влияние на выходы функциональности. Представление таблицы в виде аналитической записи – конъюнктивной нормальной формы дает потенциальную возможность существенно сократить объем диагностической информации для поиска дефектов. Последующее преобразование КНФ к ДНФ на основе тождеств алгебры логики позволяет существенно уменьшить булеву функцию.

Пример 1. Алгебрологический метод рассматривается на примере следующей таблицы неисправностей M для схемы на рис. 5 и представлен в виде пяти пунктов алгоритма.

M =

T_i / F_i	F_1	F_2	F_3	F_4	V
T_1	1				0
T_2		1			1
T_3			1		0
T_4				1	1

1. Определение всех строк таблицы неисправностей, соответствующих нулевым значениям вектора экспериментальной проверки, в целях обнуления всех единичных координат найденных строк. В данном случае – это T_1, T_4.

2. Нахождение всех столбцов, которые имеют нулевые значения координат строк с нулевым состоянием ВЭП. Обнуление единичных значений найденных столбцов.

3. Удаление из таблицы неисправностей строк и столбцов, имеющих только нулевые значения координат (найденные в пунктах 1 и 2).

M =

T_i / F_i	F_1	F_2	F_3	F_4	V
T_1	1				0
T_2		1			1
T_3			1		0
T_4				1	1

=

T_i / F_i	F_2	F_4	V
T_2	1		1
T_4		1	1

4. Построение КНФ по единичным значениям ВЭП: $F = F_2 \wedge F_4$.

5. Преобразование КНФ к ДНФ с последующей минимизацией функции. В данном случае это приводит к получению искомого результата в виде сочетания неисправностей.

Предложенный алгоритм ориентирован на анализ таблицы неисправностей в целях уменьшения ее объема и последующих вычислений, связанных с построением ДНФ, которая формирует все решения по установлению диагноза функциональностей SoC. Дальнейшему анализу подлежат функциональности F_2 и F_4.

Для этого должны быть сформированы тестовые сценарии, позволяющие проанализировать выбранные функциональности каждую по отдельности и взаимодействие между этими функциональностями. Предложенный алгоритм позволяет проверять на самом уровне операторов не всю систему в целом, а отдельно ее функциональности, в которых возникла ошибка.

Пример 2. Для таблицы M каждый тест проверяет каждую функциональность в отдельности. Однако тестовые последовательности могут быть построены таким образом, чтобы проверить одним тестом несколько функциональностей. Пример такого устройства, содержащего 6 функциональных блоков с набором из 10 тестовых сценариев, представлен в таблице неисправностей M_1.

M_1 =

T_i / F_i	F_1	F_2	F_3	F_4	F_5	F_6	V
T_1	1					1	0
T_2		1					1
T_3			1		1		1
T_4		1		1			0
T_5	1				1		0
T_6				1		1	0
T_7		1					0
T_8	1						0
T_9			1				1
T_{10}				1			1

$=$

T_i / F_i	F_2	F_3	F_4	F_5	V
T_2	1				1
T_3		1		1	1
T_9		1			1
T_{10}			1		1

Построение КНФ по единичным значениям ВЭП:

$$F = F_2 \wedge (F_3 \vee F_5) \wedge F_3 \wedge F_4 = (F_2F_3 \vee F_2F_5) \wedge F_3 \wedge F_4 = \\ = F_2F_3 \vee F_2F_3F_5 \vee F_2F_3F_4 \vee F_2F_4F_5.$$

В качестве дополнительного анализа каждого функционального блока могут быть использованы методы анализа тестопригодности графа переходов, в виде которого может быть представлена каждая функциональность устройства. Для примера 1 дополнительно должны быть проверены блоки F_2, F_4. Функциональные блоки могут быть представлены в виде двух графовых компонентов с ориентированными дугами. Такая модель предлагается взамен модели, представленной на рис. 3, поскольку автоматная модель $M = (M^{OA}, M^{CA})$ не является технологичной для пользователя при решении практических задач тестопригодного проектирования.

Графовая модель представляет собой следующую структуру:

$$\begin{cases} M = (M^{OR}, M^{CG}), \\ M^{OR} = \{R, I\}; \\ R = \{R_1, R_2, ..., R_i, ..., R_n\}, I = \{I_1, I_2, ..., I_j, ..., I_m\}; \\ R_i = f(R_k, I_j); \\ M^{CG} = \{S, T\}; \\ R = \{S_1, S_2, ..., S_i, ..., S_p\}, T = \{T_1, T_2, ..., T_j, ..., T_q\}; \\ S_i = f(S_k, T_j). \end{cases}$$

Здесь M^{OR} – граф регистровых передач С.Г Шаршунова, имеющий множество вершин R, задающих все компоненты памяти (регистры, триггеры, счетчики, память, входные и выходные шины), используемые в программе, и дуг, отмеченных инструкциями I, которые активизируют передачу информации между вершинами. Выражение $R_i = f(R_k, I_j)$ определяет функциональную зависимость между смежными вершинами $R_i \to R_k$, которые соединятся для и путем выполнения операции $I_j \in I$. Компонент M^{CG} представляется собой содержательный граф управляющего автомата, определенный на множестве вершин S, которые соединены ориентированными дугами T, отмеченными условиями переходов. Выражение $S_i = f(S_k, T_j)$ определяет функциональную зависимость между смежными вершинами $S_i \to S_k$ управляющего графа, которые соединятся для выполнения перехода $T_j \in T$.

Структура графа выглядит следующим образом:

$$G = \{R, I\};$$
$$R = \{R_1, R_2, ..., R_n\}, I = \{I_1, I_2, ..., I_m\};$$
$$I_{ij} \in I_i \approx (R_pR_q);$$

Здесь модель программного (аппаратного) модуля представлена графом $G = \{R, I\}$, состоящим из вершин (регистров) и дуг (инструкций). Каждая дуга графа отмечена не менее чем одной операцией $I_{ij} \in I_i \approx (R_pR_q)$, формирующей подмножество команд, принадлежащее дуге (R_pR_q).

Преимущества графовых моделей заключаются не только в структурном отображении взаимодействия функционалов, но и в возможности применять методы анализа тестопригодности, поскольку ориентированные графовые модели имеют выраженные направления информационных потоков, входные и выходные вершины.

3.1. Вычисление показателей тестопригодности для операционного автомата

Анализ тестопригодности основывается на вычислении для вершин количественной оценки достижимости вершин от входов (управляемость, $C(R_q)$) и от выходов (наблюдаемость, $C(R_p)$).

Вычисление управляемости. Критерий управляемости вершины $C(R_q)$ зависит от управляемости предшествующей вершины $C(R_p)$, а также от приведенной аддитивной мощности множества команд $\frac{1}{k} \times \sum_{i=1}^{k} \left[\frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_pR_q) \right| \right]$, активизирующих k дуг, которые входят в анализируемую вершину $C(R_q)$. Здесь каждая дуга графа содержит m операций, инициирующих передачу информации в (R_pR_q). По аналогии формулируется критерий оценивания наблюдаемости $C(R_p)$, ориентированный на анализ вершин преемников и дуг, исходящих из $C(R_p)$. Преимуществом введенных моделей и критериев оценивания управляемости и наблюдаемости является их универсальность, ориентированная на выполнение прямой и обратной импликации на графе, а также их инвариантность по отношению к тестопригодному анализу и синтезу тестов программных и аппаратных компонентов. Управляемость $C(R_x) = 1$ входных и наблюдаемость $O(R_y) = 1$ выходных вершин графа иницииру-

ется единичными значениями. Значение $C(R_i) = 0$ имеет вершина, которая не достижима ни по какому пути в графе. Практически значения достижимости большинства вершин находятся в границах интервала [0;1]. При безусловном переходе вес дуги равен единице.

В общем случае формула вычисления управляемости выглядит следующим образом:

$$C(R_q) = \frac{1}{k} \times \sum_{i=1}^{k} \left[\frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_p R_q) \right| \times C(R_p) \right]; \quad (4)$$

На рис. 7 представлен пример графа операционного автомата, связанные дугами вершины которого представляют собой регистры, управляемые операторами $I_1 - I_{11}$. В графе девять команд {1,2,3,4,5,6,7,8,9}, которые объединены операторами $I_1 - I_{11}$, восемь вершин – X, R0, R1, R2, R3, R4, R5, Y.

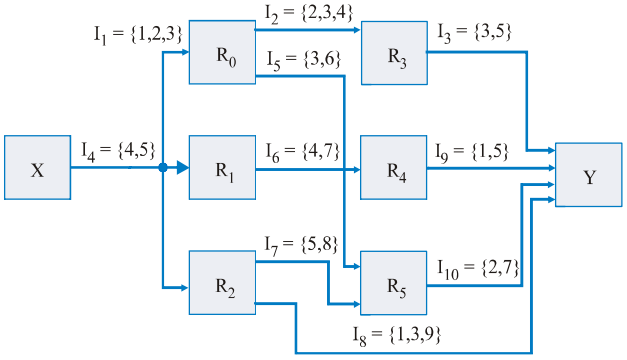


Рис. 7. Пример операционного автомата для проведения анализа тестопригодности

Показатели управляемости для графа на рис. 7 вычисляются следующим образом:

$$C(X) = 1; \quad C(R_3) = C(X) \times 1 \times \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_3 R_X) \right| = \frac{3}{9};$$

$$C(R_1) = C(X) \times 1 \times \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_1 R_X) \right| = \frac{2}{9};$$

$$C(R_2) = C(X) \times 1 \times \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_2 R_X) \right| = \frac{2}{9};$$

$$C(R_3) = C(R_0) \times 1 \times \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_3 R_0) \right| = \frac{1}{9};$$

$$C(R_4) = C(R_1) \times 1 \times \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_4 R_1) \right| = \frac{4}{81};$$

$$C(R_5) = \left(\left[C(R_0) \times \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_5 R_0) \right| \right] + \left[C(R_2) \times \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_5 R_2) \right| \right] \right) \times \frac{1}{2} = \frac{20}{81};$$

$$C(Y) = \left(\left[\frac{1}{9} \times \frac{2}{9} \right] + \left[\frac{4}{81} \times \frac{2}{9} \right] + \left[\frac{20}{81} \times \frac{2}{9} \right] + \left[\frac{2}{9} \times \frac{3}{9} \right] \right) \times \frac{1}{4} = \frac{112}{243};$$

Минимальную управляемость имеет вершина R_4.

Вычисление наблюдаемости. Наблюдаемость вершин графа вычисляется аналогично показателям управляемости. В случае вычисления наблюдаемости показатели вычисляются по направлению от выходов устройства ко входам.

Оценка наблюдаемости зависит от значения наблюдаемости предыдущей вершины, от количества операндов, активизирующих дугу. Наблюдаемость конечной вершины $O(R_Y) = 1$ или 100%. Наблюдаемость может принимать относительное значение, лежащее в интервале [0;1]. Значение $O(R_i) = 0$ имеет вершина, которая не наблюдаема ни по какому пути в графе. Практически значения наблюдаемости большинства вершин находятся в границах интервала [0;1].

Общая формула вычисления наблюдаемости выглядит следующим образом:

$$O(R_p) = \frac{1}{k} \times \sum_{i=1}^{k} \left[\frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_p R_q) \right| \times O(R_q) \right]; \quad (5)$$

Для графа на рис. 7 показатели наблюдаемости вычисляются следующим образом:

$$O(Y) = 1; \; O(R_3) = O(Y) \times \frac{2}{9} = \frac{2}{9};$$

$$O(R_4) = O(Y) \times \frac{2}{9} = \frac{2}{9}; \; O(R_5) = O(Y) \times \frac{2}{9} = \frac{2}{9};$$

$$O(R_2) = \left[\left(O(Y) \times \frac{3}{9} \right) + \left(O(R_5) \times \frac{2}{9} \right) \right] / 2 = \frac{62}{81};$$

$$O(R_1) = O(R_4) \times \frac{2}{9} = \frac{4}{81};$$

$$O(R_0) = \left[\left(O(R_3) \times \frac{3}{9} \right) + \left(O(R_5) \times \frac{2}{9} \right) \right] / 2 = \frac{20}{81};$$

$$O(X) = \left[\left(O(R_0) \times \frac{3}{9} \right) + \left(O(R_1) \times \frac{2}{9} \right) + \left(O(R_2) \times \frac{2}{9} \right) \right] / 3 = \frac{64}{81};$$

Минимальное значение наблюдаемости имеет вершина R_0.

Вычисление тестопригодности. Тестопригодность вершины в графе вычисляется по формуле:

$$T(R_i) = C(R_i) \times O(R_i). \quad (6)$$

Для схемы на рис. 7 показатели тестопригодности следующие:

$$T(X) = 1 \times 0{,}79012 = 0{,}79012;$$
$$T(R_0) = 0{,}3333(3) \times 0{,}24691 = 0{,}08230;$$
$$T(R_1) = 0{,}2222(2) \times 0{,}04938 = 0{,}01097;$$
$$T(R_2) = 0{,}2222(2) \times 0{,}7654 = 0{,}17009;$$
$$T(R_3) = 0{,}1111(1) \times 0{,}2222(2) = 0{,}2222;$$
$$T(R_4) = 0{,}04938 \times 0{,}2222(2) = 0{,}01097;$$
$$T(R_5) = 0{,}24691 \times 0{,}2222(2) = 0{,}05486;$$
$$T(Y) = 0{,}4609 \times 1 = 0{,}4609;$$

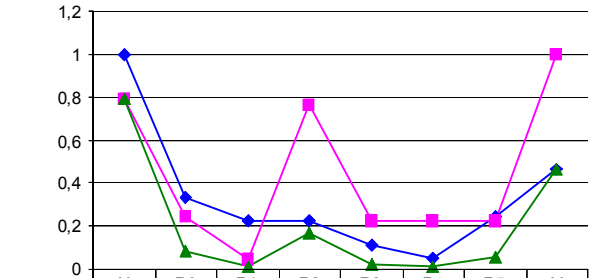
Общая тестопригодность графа вычисляется по формуле:

$$T_{total} = \frac{1}{n}\sum_{i=1}^{n} T(R_i), \qquad (7)$$

где n – количество вершин.

Для примера на рис. 7 общая тестопригодность схемы $T_{total} = 1{,}60367$.

На рис. 8 показана зависимость показателей управляемости, наблюдаемости и тестопригодности от количества вершин в графе.



	X	R0	R1	R2	R3	R4	R5	Y
C(R)	1	0,3333	0,2222	0,2222	0,1111	0,0494	0,2469	0,4609
O(R)	0,7901	0,2469	0,0438	0,7654	0,2222	0,2222	0,2222	1
T(R)	0,7901	0,0823	0,0097	0,1701	0,0247	0,011	0,0549	0,4609

Рис. 8. Показатели управляемости, наблюдаемости, тестопригодности для примера на рис. 7

По результатам анализа управляемости и наблюдаемости дополнительно должны быть проверены блоки R_1 и R_5, поскольку минимальное значение управляемости и наблюдаемости имеют вершины графа R_1 и R_5.

Пример 3. Пусть необходимо создать программное средство, позволяющее решать несложную арифметическую задачу, предложенную ниже.

Необходимо найти сумму функций $\varphi(x) + \omega(x)$, где

$$\varphi(x) = \begin{cases} x + 3, x > 2 \\ 2x - 3, 2 \le x < 12 \\ -3x + 7, x \ge 12 \end{cases};$$

$$\omega(x) = \begin{cases} \sin(x + \pi/3), x < 2\pi/3 \\ \sin(x\pi + 2), x \ge 2\pi/3 \end{cases}.$$

Ниже представлен один из возможных вариантов листинг-программы, решающей поставленную выше задачу на языке C++.

Листинг 1

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{   const double Pi=3.14159;
    double F, w, f, x;
    cin>>x;
if (x<2) f = x+3;
else if ((x>=2) && (x<12)) f=2*x-3;
else f=-3*x+7;
if (x<2./3.*Pi)
w=sin(x+Pi/3);
else w=sin(Pi*x)+2;
F=f+w;
cout<<F<<endl;
    return 0;}
```

Однако диагностируемое программное средство содержит ошибку, в результате чего конечные данные некорректны. Ошибка содержится в одном из операторов вычислительной части программы.

Листинг 2

```
w=sin(x+Pi/3);
else w=sin(Pi*x) – 2;
F=f+w;
```

Задача состоит в определении месторасположения неисправности в программном средстве на основе использования анализа тестопригодности граф-модели. Рассматриваемому программному средству соответствует следующий граф регистровых передач (рис. 9):

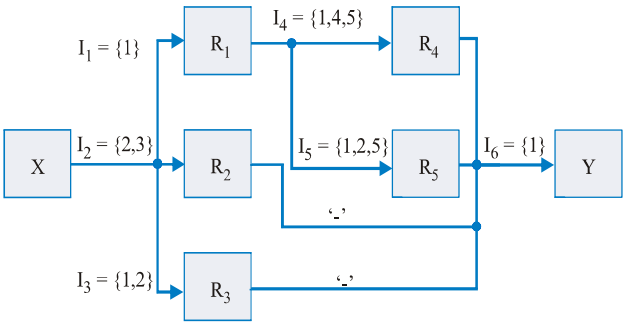


Рис. 9. Граф регистровых передач

В предложенном графе на каждой из дуг выполняются некоторые последовательные операции Ii. В скобках указывается, какие именно операции (в данном примере исключительно арифметические) выполняются

с исходными данными. На номер операции указывает соответствующая цифровая метка:

{1} – операция суммирования “+”;

{2} – операция произведения “*”;

{3} – операция вычитания “-”;

{4} – операция деления “/”;

{5} – нахождение тригонометрического синуса “sin”.

По частоте встречаемости операций в коде можно выделить операции суммирования (пять раз), произведения (четыре раза), деления (два раза), нахождение тригонометрического синуса “sin” (два раза). Предположительно, неисправность может возникнуть в строке кода, где производятся операции суммирования {1} и умножения {2}.

Для определения дополнительно проверяемых строчек кода необходимо провести анализ тестопригодности.

Показатели управляемости следующие:

$$C(X)=1;\quad C(R_1)=C(X)\times 1\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(R_3R_X)\right|=\frac{1}{5};$$

$$C(R_2)=C(X)\times 1\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(R_1R_X)\right|=\frac{2}{5};$$

$$C(R_3)=C(X)\times 1\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(R_2R_X)\right|=\frac{2}{5};$$

$$C(R_4)=C(R_1)\times 1\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(R_3R_0)\right|=\frac{3}{25};$$

$$C(R_5)=C(R_1)\times 1\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(R_4R_1)\right|=\frac{3}{25};$$

$$C(Y)=\left(\begin{array}{l}\left[C(R_4)\times\frac{1}{5}\right]+\left[C(R_2)\times\frac{1}{5}\right]+\\ +\left[C(R_3)\times\frac{1}{5}\right]+\left[C(R_5)\times\frac{1}{5}\right]\end{array}\right)\times\frac{1}{4}=\frac{13}{250};$$

Показатели наблюдаемости:

$$O(Y)=1;\ O(R_5)=O(Y)\times\frac{1}{5}=\frac{1}{5};$$

$$O(R_4)=O(Y)\times\frac{1}{5}=\frac{1}{5};\quad O(R_2)=O(Y)\times\frac{1}{5}=\frac{1}{5};$$

$$O(R_3)=O(Y)\times\frac{1}{5}=\frac{1}{5};$$

$$O(R_1)=\left[\left(O(R_4)\times\frac{3}{5}\right)+\left(O(R_5)\times\frac{3}{5}\right)\right]/2=\frac{20}{81};$$

$$O(X)=\left[\left(O(R_1)\times\frac{1}{5}\right)+\left(O(R_2)\times\frac{2}{5}\right)+\left(O(R_3)\times\frac{2}{5}\right)\right]/3=\frac{26}{375};$$

Показатели тестопригодности следующие:

$$T(X)=0{,}069;\quad T(R_1)=0{,}024;$$
$$T(R_2)=0{,}08;\quad T(R_3)=0{,}08;$$
$$T(R_4)=0{,}0{,}024;\quad T(R_5)=0{,}024;$$
$$T(Y)=0{,}052;$$

При выборе контрольных точек входные и выходные вершины не принимаются во внимание, так как являются стопроцентно наблюдаемыми и управляемыми. Из анализа тестопригодности видно, что минимальную управляемость имеет вершина R_1, которая активизируется операцией {1} – операцией сложения. Минимальную наблюдаемость имеют вершины R_2, R_3, R_4, R_5 с набором команд {2,3}, {1,2}, {1,4,5}, {1,2,5} соответсвенно. Минимальную тестопригодность имеют вершины R_1, R_4, R_5, которые активизируются командами {1}, {1,4,5}, {1,2,5}. Исходя из этого, видно, что должны быть проверены операции сложения, деления и нахождение тригонометрического синуса “sin”.

В коде из примера это следующие строки, одна из которых и содержала ошибку:

```
w=sin(x+Pi/3);
else w=sin(Pi*x)+2.
```

4. Вычисление показателей тестопригодности для управляющего автомата

Здесь основная сложность тестирования заключается в том, что в FSM должны быть проверены все трудно достижимые состояния; тупиковые ситуации или коллизии двух потоков данных; места локализации ветвлений и обратных связей в коде (if, case, loop). Относительно управляющих автоматов можно выделить следующее правило: каждый FSM должен содержать ассерции, которые проверяют кодирование состояний и переходов [19]. Примеры использования ассеций для верификации FSM:

assert_bits, assert_next_state, assert_value, assert_cycle_sequence, assert_code_distance, assert_next, assert_transition, assert_one_hot, assert_one_cold, assert_zero_one_hot.

Предлагаемый в работе метод заключается в вычислении значений управляемости и наблюдаемости вершин, формирующих оценку тестопригодности.

4.1. Вычисление показателей тестопригодности для управляющего автомата

Вычисление управляемости. Показатели управляемости для графа на рис. 5 вычисляются следующим образом:

$$C(S_0)=1;\quad C(S_3)=C(S_0)\times 1\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(S_0S_3)\right|=\frac{3}{4};$$

$$C(S_5)=C(S_3)\times 1\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(S_3S_5)\right|=\frac{6}{16};$$

$$C(S_4)=\left(\left[C(S_1)\times\frac{1}{4}\times 1\right]+\left[C(S_3)\times\frac{1}{4}\times 1\right]+\left[C(S_5)\times\frac{1}{4}\times 1\right]\right)\times\frac{1}{3}=\frac{47}{384};$$

$$C(S_2)=\left(\left[C(S_1)\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(S_1S_2)\right|\right]+\left[C(S_4)\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(S_2S_4)\right|\right]\right)\times\frac{1}{2}=\frac{179}{1536};$$

$$C(S_1)=\left(\left[C(S_0)\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(S_0S_1)\right|\right]+\left[C(S_3)\times\frac{1}{m}\times\left|\bigcup_j I_{ij}\in(S_0S_3)\right|\right]\right)\times\frac{1}{2}=\frac{11}{32};$$

Минимальную управляемость имеет вершина S_2.

Вычисление наблюдаемости. Для графа на рис. 10 показатели наблюдаемости вычисляются следующим образом:

$$O(S_2)=1;\quad O(S_4)=O(S_2)\times 1\times\frac{2}{4}=\frac{1}{2};$$

$$O(S_5)=O(S_4)\times 1\times\frac{1}{4}=\frac{1}{8};$$

$$O(S_3)=\left[\left(O(S_1)\times\frac{1}{4}\right)+\left(O(S_4)\times\frac{1}{4}\right)+\left(O(S_5)\times\frac{2}{4}\right)\right]\times\frac{1}{3}=\frac{7}{96};$$

$$O(S_1)=\left[\left(O(S_2)\times\frac{2}{4}\right)+\left(O(S_4)\times\frac{1}{4}\right)\right]\times\frac{1}{2}=\frac{1}{8};$$

$$O(S_0)=\left[\left(O(S_1)\times\frac{2}{4}\right)+\left(O(S_3)\times\frac{3}{4}\right)\right]\times\frac{1}{2}=\frac{15}{256};$$

Минимальное значение наблюдаемости имеет вершина S_0.

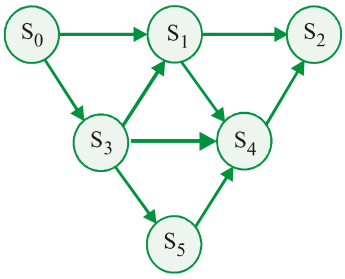


Рис. 10. Пример графа управляющего автомата

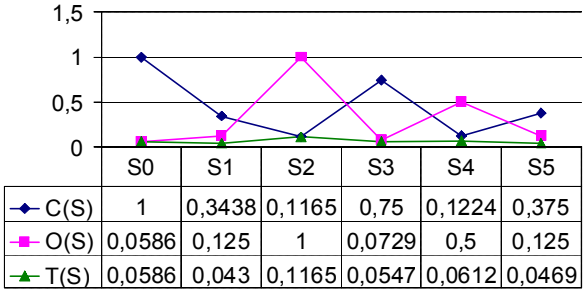
Вычисление тестопригодности.

$$T(S_0)=1\ \times\ 0{,}05859375=0{,}05859375;$$
$$T(S_1)=0{,}34375\ \times\ 0{,}125=0{,}04296875;$$
$$T(S_2)=0{,}11654\ \times\ 1=0{,}11654;$$
$$T(S_3)=0{,}75\ \times\ 0{,}07292=0{,}05469;$$
$$T(S_4)=0{,}1224\times\ 0{,}5=0{,}0612;$$
$$T(S_5)=0{,}375\ \times\ 0{,}125=0{,}046875;$$

Наихудшую тестопригодность имеет вершина S_4.

Общая тестопригодность схемы $T_{total}=0{,}063478$.

На рис. 11 представлены показатели управляемости, наблюдаемости и тестопригодности для каждой вершины графа.



	S0	S1	S2	S3	S4	S5
C(S)	1	0,3438	0,1165	0,75	0,1224	0,375
O(S)	0,0586	0,125	1	0,0729	0,5	0,125
T(S)	0,0586	0,043	0,1165	0,0547	0,0612	0,0469

Рис. 11. Показатели тестопригодности

5. Стратегия выбора контрольных точек и введения ассерций в код устройства

Стратегия выбора точек для модификации графа состоит в следующем: выбираются 3% линий с минимальными значениями достижимости A. При этом к выбранным линиям добавляются другие, имеющие оценки, равные максимальному из 3% выбранных, если таковые имеются. Обычно линий с одинаковыми значениями показателей сравнительно мало – это особенность метода анализа тестопригодности. 3%-ная квота линий была выбрана из накладываемых ограничений на количество внешних дополнительных контактов в устройстве – не более 5%.

Далее с полученным множеством пересекается множество контрольных точек в коде, выбранных на основе общепринятых правил установки ассерций в коде.

В [19] представлен набор правил, предположений и рекомендаций по использованию ассерций. Всего таких правил порядка сорока. Например, для устройств АЛУ ассерции должны проверять правильность выполнения операций, переполнение разрядной сетки и др. Каждый модуль памяти, регистры FIFO, LIFO должны быть проверены ассерциями.

Формула, по которой в коде устройства могут быть выбраны контрольные точки для внедрения ассерций, выглядит следующим образом:

$$Z=\{Y_{TY}\}\cap\{A_{Rules}\}/\{Z_{TY}\},$$

где $\{Y_{TY}\}$ – множество точек, выбранных по методу анализа тестопригодности; $\{A_{Rules}\}$ – множество контрольных точек в устройстве, выбранных на основе правил использования ассерций в коде устройства; $\{Z_{TY}\}$ – множество точек в устройстве, которые были выбраны по методу анализа тестопригодности, но не могут быть использованы в качестве ассерций.

Модификация структуры графа. При выполнении процедуры обхода всех вершин графа и выбора минимального пути для построения теста возникает про-

блема достижимости состояний. Из-за наличия дополнительных внутренних переменных в функции возбуждения полученный путь может не существовать для содержательного графа. В этом случае необходима модификация пути. Предложенная стратегия модификации состоит в разделении режимов тестирования и нормального функционирования схемы. Для этого на каждую выбранную по стратегии вершину в графе ставится условная вершина, которая предполагает 100% управляемость выбранной вершины графа. Ожидается, что такой подход к модификации устройства позволит при небольших аппаратурных затратах существенно повысить тестопригодность разрабатываемого устройства.

6. Заключение

Научная новизна: разработан метод расчета показателей тестопригодности, а также стратегии выбора точек для модификации устройства и способ модификации содержательного графа FSM в целях увеличения его тестопригодности и повышения качества покрытия неисправностей тестом.

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

1. Представлена модифицированная модель операционного устройства С.Г. Шаршунова в виде композиции управляющего и операционного автомата.

2. Разработан метод анализа тестопригодности (управляемости и наблюдаемости) программных и аппаратных продуктов, модели которых представлены ориентированными графами регистровых передач и управления.

3. Разработана технологическая модель ассерций для программного продукта, широко используемая для тестирования и верификации цифровых систем на кристаллах. Ассерции размещаются внутри моделей устройств с возможностью анализа значений всех внутренних сигналов, что дает возможность легко определить место дефекта, а также уменьшить время обнаружения неисправностей при проектировании.

4. Предложена методика внедрения ассерций в функциональные блоки устройства на основе вычисленных показателей управляемости и наблюдаемости.

Практическая значимость предложенных методик и моделей заключается в высокой заинтересованности софтверных компаний в инновационных решениях проблемы эффективного тестирования и верификации программных продуктов, предложенных выше.

Преимущества предложенных методов и алгоритмов:

1. Применение модели ассерций сокращает объем кода для построения testbench, а значит, существенно сокращает время ручного проектирования квалифицированного инженера в 2 – 3 раза, которое является наиболее дорогостоящим компонентом любого проекта. Кроме того, сокращение объема кода в модели ассерций в несколько раз уменьшает время моделирования проекта и вероятность возникновения в нем ошибок [14].

2. В настоящее время места локализации ассерций выбираются инженером на свое усмотрение. Использование разработанных методов анализа тестопригодности позволяет создать алгоритм выбора узких мест в устройстве для дальнейшего внедрения ассерций, что существенно сократит временные затраты на выбор блоков в устройстве, которые должны быть дополнительно покрыты ассерциями.

3. При вычислении показателей тестопригодности нет необходимости вычислять систему уравнений при наличии обратных связей, как для ранее разработанных методов [32-35].

Недостатком предложенного метода можно считать то, что анализ на вентильном уровне дает более точные показатели тестопригодности для дальнейшей модификации схемы, чем на более высоких уровнях описания устройства. Также в схему вводится избыточность в виде дополнительных вершин и дуг, что приводит к появлению дополнительных переменных.

Литература: 1. *IEEE* P1500/D11. January 2005. Draft Standard Testability Method for Embedded Core-based Integrated Circuits. New York. 2005. 138 p. **2.** *Abramovici M., Breuer M.A. and Friedman A.D*. Digital systems testing and testable design. Computer Science Press. 1998. 652 p. **3.** *IEEE* Std 1149.1-2001. Standard Test Access Port and Boundary-Scan Architecture. New York. 2001. 208 p. **4.** *IEEE* Std 1149.4-1999. IEEE Standard for a Mixed-Signal Test Bus. New York, 2000. 84 p. **5.***IEEE* Std 1149.6-2003. Standard for Boundary-Scan Testing of Advanced Digital Networks. New York. 2003. 139 p. **6.** *Jutman A*. Shift Register Based TPG for At-Speed Interconnect BIST. *Proc. of 24th International Conference on Microelectronics (MIEL'04)* // Serbia and Montenegro. 2004. P. 751-754. **7.** *Su C.,Jeng S.W.,Chen Y.T.* Boundary scan BIST methodology for reconfigurable systems. *Proc. ITC'98*. P. 774-783. **8.** *Feng W., Meyer F.J., Lombardi F.* Novel control pattern generators for interconnect testing. *Proc. Int'l Symp. Defect and Fault Tolerance in VLSI Systems*. 1999. P. 112-120. **9.** *Jin L.* The driver/receiver conflict problem in interconnect testing with boundary-scan. Proc of Asian Test Symposium (ATS'93). Beijing. 1993. P. 210-214. **10.** *Hassan A., Agarwal V., Nadeau-Dostie B., and Rajski J.* BIST of PCB interconnects using boundaryscan architecture. *IEEE Trans. Computer-Aided Design*. Vol. 11. P. 1278–1287. **11.** *Ulf Pillkahn.* Structural test in a board self test environment. Proc. IEEE Int. Test Conf. (ITC'2000). Atlantic City. NJ. USA. 2000. P. 1005-1012. **12.** *W.H. Kautz.* Testing of Faults in Wiring Interconnects. *IEEE Trans. Computers*. Vol. 23, No. 4. 1974. P. 358-363. **13.** *Stephen Pateras.* IP for Embedded Diagnosis. IEEE Design and Test of Computers. 2002. P. 46-56. **14.** *S.G. Hemmady, T.L. Anderson, Y. Zorian.* Verification and Testing of Embedded Cores. Proc. Design SuperCon. On-Chip Design. 1997. P 119-122. **15.** *Samvel Shoukourian.* A Unified methodology for Offline and Online Testing. IEEE Design & Test of Computers. 1998. P. 73-79. **16.** *Шаршунов С.Г.* Построение тестов микропроцессоров. 1. Общая модель. Проверка обработки данных//Автоматика и телемеханика. 1985. №11. С.145-155. **17.** *Чипулис В.П., Шаршунов С.Г.* Построение тестов микропроцессоров. 2. Проверка хранения и передачи данных

// Автоматика и телемеханика. 1986. №1. С.139-150. **18.** *Advanced* Verification Methodology Cookbook, Version 2.0, Mark Glasser, Adam Rose, Tom Fitzpatrick, Dave Rich, Harry Foster, July 24, 2006. **19.** *Verification* Methodology Manual for System Verilog, Janick Bergeron, Eduard Cerny, Alan Hunter, Andrew Nightingale, 2006 Synopsys, Inc. and ARM Limited. P. 528. **20.** *OVM* Class Reference, Version 1.0.1, February 2008. 286 p. **21.** *Miro Samek,* C/C++ Users Journals, advanced Solutions for professional developers, August 2003. 8p. **22.** *Performance* Assertions for Mobile Devices, Raimondas Lencevicius, Edu Metz, *ISSTA'06,* July 17–20, 2006, Portland, Maine, USA. **23.** *Synthesis* of Synchronous Assertions with Guarded Atomic Actions, Michael Pellauer, Mieszko Lis, Donald Baltus, Rishiyur Nikhil, 15th March 2005. **24.** *Being* Assertive With Your X (SystemVerilog Assertions for Dummies), Don Mills, SNUG San Jose 2004. **25.** *Armoni R., Fix L., Flaisher A., Gerth R., Ginsburg B., Kanza T., Landver A., Mador-haim S., Singerman E., Tiemeyer A., Vardi M. Y., and Zbar Y.* The ForSpec Temporal Logic: A new temporal property-specification language. In Tools and Algorithms for Construction and Analysis of Systems // Lecture Notes in Computer Science. 2002. Vol. 2280. P. 211–296. **26.** *Eisner C., Fisman D., Havlicek J., McIsaac, and Van Campenhout D.* The definition of a temporal clock operator // Proc. ICALP. Eindhoven: Springer-Verlag, July 2003. P. 857-870. **27.** *Beer I., Ben-David S., Eisner C., and Landver A.* RuleBase: An industry-oriented formal verification tool // Proc. Design Automation.– New York: ACM Press, 1996. P. 655–660. **28.** *Arons T., Elster E., Fix L., Mador-Haim S., Mishaeli M., Shalev J., Singerman E., Tiemeyer A., Vardi M. Y., Zuck L. D.* Formal Verification of Backward Compatibility of Microcode // Computer Aided Verification. 2005. Vol. 4, № 1. P. 185-198. **29.** *Кулак Э.Н., Каминская М.А., Ваде Гриби, Хассан Ктейман, Гузь О.О.* Модификация цифровых схем с использованием метода анализа тестопригодности TADATPG // Радиоэлектроника и информатика. 2005. №4. С. 60-68. **30.** *Хаханов В.И., Ковалев Е.В., Ханько В.В., Масуд М.Д. Мехеди.* Система генерации тестов для проектирования цифровых автоматов в среде Active-HDL // АСУ и приборы автоматики. 2000. Вып. 111.С. 15-22. **31.** *Bayraktaroglu Ismet, Orailoglu Alex* (2005) The Construction of Optimal Deterministic Partitionings in Scan-Based BIST Fault Diagnosis: Mathematical Foundations and Cost-Effective Implementations. IEEE Transactions on Computers. P. 61-75. **32.** *Rutman R. A.* Fault Detection Test Generation for Sequential Logic Heuristic Tree Search // IEEE Computer Repository Paper No. R-72-187, 1972. **33.** *Grason J.* TMEAS - A Testability Measurement Program // Proc. 16th Design Automation Conf., pp. 156-161, June, 1979. **34.** *Grason J. and Nagel A. W.* Digital Test Generation and Design for Testability // Journal Digital Systems, 1981.Vol.5. No. 4. P. 319-359. **35.** *Parker K.P., McCluskey E.J.* Probabilistic Treatment of General Combinational Networks // IEEE Trans. on Computers. Vol. C-24. no. 6. 1975. P. 668–670.

Поступила в редколлегию 31.01.2008

Рецензент: д-р техн. наук, проф. Кривуля Г.Ф.

Хаханов Владимир Иванович, декан факультета компьютерной инженерии и управления ХНУРЭ, д-р техн. наук, профессор кафедры автоматизации проектирования вычислительной техники, IEEE Computer Society Golden Core Member. Научные интересы: проектирование и диагностика цифровых систем, сетей и программных продуктов. Увлечения: футбол, баскетбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (057) 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Каминская Марина Александровна, аспирантка кафедры АПВТ ХНУРЭ, инженер по тестированию компании ИП «Интспеи-Украина». Научные интересы: тестопригодное проектирование, моделирование и верификация цифровых систем на кристаллах. Увлечения: литература, музыка. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 7021326. E-mail: maryna4329@kture.kharkov.ua.

Сушанов Алексей Владимирович, студент ХНУРЭ, группа КИ-07-6. Научные интересы: программирование, проектирование цифровых систем на кристаллах. Увлечения: литература. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 7021326. E-mail: hahanov@kture.kharkov.ua.

УДК681.3

РАСШИРЕНИЕ ФОРМАТА МИКРОКОМАНД В МИКРОПРОГРАММНОМ УСТРОЙСТВЕ УПРАВЛЕНИЯ С РАЗДЕЛЕНИЕМ КОДОВ

БАРКАЛОВ А.А., ТИТАРЕНКО Л.А., МИРОШКИН А.Н.

Предлагается метод синтеза композиционных микропрограммных устройств управления с разделением кодов. Метод направлен на уменьшение количества макроячеек в комбинационной части устройства и основан на расширении формата микрокоманд полем кода класса псевдоэквивалентных ОЛЦ. Приводятся условия и результаты исследований применения данного метода.

1. Введение

При проектировании цифровых управляющих устройств (УУ) одной из актуальных задач является соблюдение ограничений на аппаратурные затраты в его схеме [1]. Для реализации линейных алгоритмов управления, доля операторных вершин в которых больше 75%, целесообразно использовать УУ класса композиционных микропрограммных устройств управления (КМУУ) [2]. В настоящее время базис ПЛИС (программируемых логических интегральных микросхем) [3,4] широко используется для реализации схем устройств управления [5,6]. Проблема минимизации аппаратурных затрат решается путем преобразования исходных дизъюнктивных нормальных форм (ДНФ) функций автомата в целях минимизации числа термов в них, а следовательно, и необходимого для их реализации количества макроячеек программируемой матричной логики (ПМЛ). В настоящей работе предлагается один из путей решения этой проблемы: реализация управляющего устройства в виде КМУУ с разделением кодов [2] и расширением формата микрокоманд.

Целью исследования является оптимизация комбинационной схемы КМУУ за счет введения в формат микрокоманд кодов классов псевдоэквивалентных операторных линейных цепей (ПОЛЦ).