МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ
УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

ISSN 0135-1710

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ

**Всеукраинский межведомственный
научно-технический сборник**

Основан в 1965 г.

Выпуск 145

Харьков
2008

УДК 62-50

В сборнике представлены результаты исследований, касающихся компьютерной инженерии, управления, технической диагностики, автоматизации проектирования, оптимизированного использования компьютерных сетей и создания интеллектуальных экспертных систем. Предложены новые подходы, алгоритмы и их программная реализация в области автоматического управления сложными системами, оригинальные информационные технологии в науке, образовании, медицине.

Для преподавателей университетов, научных работников, специалистов, аспирантов.

У збірнику наведено результати досліджень, що стосуються комп'ютерної інженерії, управління, технічної діагностики, автоматизації проектування, оптимізованого використання комп'ютерних мереж і створення інтелектуальних експертних систем. Запропоновано нові підходи, алгоритми та їх програмна реалізація в області автоматичного управління складними системами, оригінальні інформаційні технології в науці, освіті, медицині.

Для викладачів університетів, науковців, фахівців, аспірантів.

Редакционная коллегия:

В.В. Семенец, д-р техн. наук, проф. (гл. ред.), *М.Ф. Бондаренко,* д-р техн. наук, проф., *И.Д. Горбенко,* д-р техн. наук, проф., *Е.П. Путятин,* д-р техн. наук, проф., *В.П. Тарасенко,* д-р техн. наук, проф., *Г.И. Загарий,* д-р техн. наук, проф., *А.Штефан,* доктор-инженер, *Г.Ф. Кривуля,* д-р техн. наук, проф., *О.Г. Руденко,* д-р техн. наук, проф., *Н.В. Алипов,* д-р техн. наук, проф., *Е.В. Бодянский,* д-р техн. наук, проф., *Э.Г. Петров,* д-р техн. наук, проф., *В.Ф. Шостак,* д-р техн. наук, проф., *В.М. Левыкин,* д-р техн. наук, проф., *В.И. Хаханов,* д-р техн. наук, проф. (отв. ред.).

Свидетельство о государственной регистрации
печатного средства массовой информации КВ № 12073-944ПР от 07.12.2006 г.

Адрес редакционной коллегии: Украина, 61166, Харьков, просп. Ленина, 14, Харьковский национальный университет радиоэлектроники, комн. 321, тел. 70-21-326

© Харківський національний університет
радіоелектроніки, 2008

СОДЕРЖАНИЕ

ЕВЛАНОВ М.В., СЛИПЧЕНКО Е.В., НИКИТЮК В.А.
ПОДХОД К ПРОЕКТИРОВАНИЮ ХРАНИЛИЩ ДАННЫХ ВИНФОРМАЦИОННЫХ СИСТЕМАХ 4
БАБИЧ А.В., КУДИНА М.В., ЕМЕЛЬЯНОВ И.В.
ИССЛЕДОВАНИЕ МЕТОДОВ РЕШЕНИЯ ЗАДАЧ ИНЖИНИРИНГА ТРАФИКА ВСЕТЯХ СЛЕДУЮЩЕГО ПОКОЛЕНИЯ 8
ЗАЙЧЕНКО С.А., ХАХАНОВ В.И.
ФОРМАЛЬНАЯ СЕМАНТИКА СЛОЖНЫХ ОПЕРАТОРОВ ЛИНЕЙНОЙ ТЕМПОРАЛЬНОЙ ЛОГИКИ 14
ОЛІЙНИК О.І.
ЧИСЕЛЬНЕ МОДЕЛЮВАННЯ ДИФУЗІЙНИХ ПРОЦЕСІВ У СИСТЕМАХ З МАСИВАМИ ЗАГЛИБЛЕНИХ МІКРОЕЛЕКТРОДІВ У ФОРМІ ДИСКУ 29
ЛИТВИНОВА Е.И.
ТЕХНОЛОГИИ ВСТРОЕННОГО РЕМОНТА КОМПОНЕНТОВ SYSTEM-IN-PACKAGE. 40
ТЕВЯШЕВ А.Д., ЗОЛОТАРЕВ Д.А.
ОБ ОДНОМ МЕТОДЕ РЕШЕНИЯ ЗАДАЧИ ОПТИМИЗАЦИИ ПЛАНОВЫХ РЕЖИМОВ ТРАНСПОРТА И РАСПРЕДЕЛЕНИЯ ПРИРОДНОГО ГАЗА ВГОРОДСКИХ ГАЗОРАСПРЕДЕЛИТЕЛЬНЫХ СЕТЯХ 48
ЧУБ И.А., НОВОЖИЛОВА М.В.
МОДИФИКАЦИЯ ТОЧНОГО МЕТОДА РЕШЕНИЯ ЗАДАЧИ РАЗМЕЩЕНИЯ ПРЯМОУГОЛЬНЫХ ОБЪЕКТОВ 57
ТАЯНОВ С.А., ТАЯНОВ В.А.
МЕТОДИКА КЛАСТЕРИЗАЦІЇ ЗОБРАЖЕНЬ ДЛЯ ЇХ КОМПРЕСІЇ НА ОСНОВІ КОМПОНЕНТНОГО АНАЛІЗУ 63
ШАХОВСЬКА Н.Б., УГРИН Д.І.
ТЕХНОЛОГІЯ ETL В ІНТЕГРАЦІЇ ДАНИХ ТУРИСТИЧНОГО БІЗНЕСУ 68
ДОЛГОВА Н.Г., НОВОЖИЛОВА М.В., СИНЕЛЬНИКОВА О.И.
МЕТОД ОЦЕНКИ АЛЬТЕРНАТИВНЫХ ВАРИАНТОВ ФУНКЦИОНАЛЬНОГО ЗОНИРОВАНИЯ ТЕРРИТОРИИ ГОРОДА 73
ЕВГРАФОВ В.Н.
ПРОИЗВОДИТЕЛЬНОСТЬ БУФЕРНЫХ АСИНХРОННЫХ МНОГОСТУПЕНЧАТЫХ СЕТЕЙ С ПРОИЗВОЛЬНЫМ ЧИСЛОМ ПРИОРИТЕТНЫХ МОДУЛЕЙ ПАМЯТИ 80
КАКУРИН Н.Я., КОВАЛЕНКО С.Н., ЛОПУХИН Ю.В., МАКАРЕНКО А.Н.
СПОСОБ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ ПРЕОБРАЗОВАТЕЛЕЙ КОДОВ НА СЧЕТЧИКАХ 86
ЗАЯЦЬ В.М.
ПЕРСПЕКТИВИ ЗАСТОСУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ТА ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ КОМП'ЮТЕРА, ОПИСАНОЇ НА ОСНОВІ ДИСКРЕТНОЇ МОДЕЛІ 96
ЛЫСЕНКО Э.В., ПОНОМАРЕНКО В.П., ПИСКЛАКОВА В.П.
СИСТЕМОЛОГИЧЕСКИЙ АНАЛИЗ ПРОБЛЕМЫ ПРИНЯТИЯ РЕШЕНИЙ В УСЛОВИЯХ МНОГОКРИТЕРИАЛЬНОСТИ И НЕОПРЕДЕЛЕННОСТИ 104
ВИШНЯК М.Ю., ДОВГАНЬ С.С.
РАСШИРЕНИЕ ФУНКЦИЙ АСУ: УПРАВЛЕНИЕ ЗНАНИЯМИ 109
ОМРИ КАРИМ
ПОСТРОЕНИЕ АЛГОРИТМА РАСПОЗНАВАНИЯ ОТПЕЧАТКОВ ПАЛЬЦЕВ ДЛЯ СИСТЕМЫ КОНТРОЛЯ ДОСТУПА 116
КРИВУЛЯ Г.Ф., РЯБЕНЬКИЙ В.М., ПЕТРЕНКО Л.П.
МЕТОДИКА ФОРМИРОВАНИЯ МАТЕМАТИЧЕСКОЙ МОДЕЛИ СУММАТОРА В ФОРМАТЕ ТРОИЧНОЙ СИСТЕМЫ СЧИСЛЕНИЯ 121
КУЗЁМИН А. Я., ГОЛОВИЙ (ГУСАРЬ) Н.В., ДАЮБ Я.
РЕАЛИЗАЦИЯ МОДЕЛИ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ В ОБЛАСТИ СЕРВИСНОГО ОБСЛУЖИВАНИЯ БАНКОМАТОВ 134
ВЕРЕЩАК И.А.
МЕТОД НИЗКОУРОВНЕВОЙ ОБРАБОТКИ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ 139

РЕФЕРАТИ 143

УДК 681.326:519.613

С.А. ЗАЙЧЕНКО, В.И. ХАХАНОВ

ФОРМАЛЬНАЯ СЕМАНТИКА СЛОЖНЫХ ОПЕРАТОРОВ ЛИНЕЙНОЙ ТЕМПОРАЛЬНОЙ ЛОГИКИ

Рассматривается проблема нечеткости интерпретации сложных операторов линейной темпоральной логики в применении к анализу цифровых систем в рамках динамических методов верификации. Вводятся формальные определения и спецификации ключевых вычислительных процедур рабочего цикла верификации, ориентированные на максимальное быстродействие анализа.

1. Введение

Любая система на кристалле является параллельной. В общем случае под параллельными понимаются программно-аппаратные системы с несколькими параллельными процессами, которые непрерывно обмениваются сообщениями с внешней средой. Большинство цифровых аппаратных систем имеют конечное число состояний, определяющееся набором достижимых значений всех сигналов. В отличие от аппаратных систем, большинство программных систем, где применяются динамические структуры данных, имеют бесконечное число состояний, что значительно усложняет процессы их верификации [1,2].

Задача верификации параллельных систем состоит в доказательстве наличия или отсутствия в их работе определенных последовательностей состояний, называемых ограничениями. Если число состояний конечно, представляется возможным формальное доказательство удовлетворения системой заданных ограничений.

Очевидно, анализ работы параллельных систем, также как и анализ ограничений, предполагает формальные рассуждения об изменении состояния системы с течением времени. Формальное утверждение о том, что система всегда или в выбранном интервале времени удовлетворяет заданному логико-временному ограничению последовательностей состояний, называется темпоральным утверждением или темпоральной *ассерцией*. Необходимы формальные механизмы для моделирования параллельных систем и механизмы описания ассерций.

Темпоральная логика позволяет задать порядок событий во времени без внесения понятия времени в явном виде. Применение темпоральной логики для описания ограничений параллельных систем показало свою эффективность при специфицировании требований и верификации цифровых систем.

С точки зрения цели специфицирования различают два принципиальных вида темпоральных ограничений, которые можно задавать при помощи утверждений темпоральной логики [2]:

– *инвариантные* ограничения (safety) – заданная последовательность состояний никогда не случится (например, ресурс никогда не будет использован двумя процессами одновременно);

– *достижимые* ограничения (liveness) – желаемая последовательность состояний будет достигнута (например, пришедший запрос когда-нибудь обязательно будет обработан).

Помимо цели, темпоральные ограничения могут дополнительно характеризоваться *требованиями допустимости* (fairness constraints). Допустимость является требованием к множеству разрешенных в системе вычислительных путей. Характеристика допустимости является важным фактором для путей бесконечной длины, возникающих при циклических вычислительных процедурах. Различают три вида требования допустимости:

– *безусловная* допустимость (unconditional fairness) – для всех переходов независимо от их вхождения в состав выбранного пути каждый переход выполняется бесконечно часто;

– *слабая условная* допустимость (weak fairness) – некоторые из переходов, включенных в состав выбранного пути, будут выполнены бесконечно часто:

– *сильная условная* допустимость (strong fairness) – все переходы, включенные в состав выбранного пути, будут выполнены бесконечно часто.

Существует два основных подхода к решению задачи верификации цифровых систем на основе темпоральных ассерций: формальные методы [3] (математическое доказательство удовлетворения системой инвариантных и достижимых ограничений без тестовых воздействий) и динамические методы [4] (проверка выполнения системой заданных ограничений во время имитационного моделирования, или симуляции, под воздействием тестовых стимулов). Формальные методы гарантируют 100% точность результатов верификации, однако их применение сопряжено со значительными вычислительными затратами в связи с экспоненциальной сложностью. На практике, несмотря на отсутствие 100% гарантии результатов, более распространено применение динамических методов, более приспособленных к проверке работоспособности системы в основных операционных режимах.

Рассуждения о бесконечных вычислительных путях в рамках процесса динамической верификации системы имеют условный характер. Разумеется, число шагов верификации в системе с тестовыми воздействиями определяется длиной теста, и не может идти речи о фактической бесконечности вычислительного пути. В этом контексте ограничения допустимости следует интерпретировать относительно момента завершения тестирования.

Типичная математическая база анализа ассерций во время симуляции, состоящая в трансформации операторов LTL-логики к детерминированным конечным автоматам, лишь частично способна удовлетворить основные требования к верификации на основе ассерций [5]. Основной проблемой классической модели является экспоненциальный рост количества состояний в зависимости от структурной сложности реализуемой темпоральной формулы. Кроме того, накладывается существенное ограничение на языки описания ассерций в виде простого подмножества, поскольку классическая модель неспособна осуществить проверку ряда темпоральных конструкций с приемлемой для практического применения вычислительной сложностью. Поддержка режима глобального времени практически невозможна на основе модели конечных автоматов.

Очевидно, для обеспечения существенно лучшей производительности анализа ассерций и поддержки темпоральных операторов, определяемых для бесконечных вычислительных путей, необходима принципиально иная модель. Такими свойствами обладает ранее предложенная модель динамических регистровых очередей (DRTLQ – Dynamic Register-Transfer-Level Queues) [6]. Модель ориентирована на высокопроизводительный анализ элементов линейной темпоральной логики и эффективность обнаружения и локализации нарушений [7]. Ключевые понятия, вводимые в модели (последовательностные функции, темпоральные свойства), представляют собой обобщенную надстройку над общеизвестными терминами LTL и языков описания ассерций [8].

Цель данной работы – формальное определение семантики сложных операторов линейной темпоральной логики в контексте их реализации в модели DRTLQ, осуществляющей высокопроизводительный анализ ассерций в рамках динамических методов верификации цифровых систем, а также создание формальной спецификации вычислительных процедур для наиболее сложных элементов модели DRTLQ.

К основным *задачам* исследования относятся:

– уточнение ранее предложенной модели процесса анализа темпоральных формул в рамках динамических методов верификации (под управлением HDL-симулятора);

– формальное определение семантики сложных операторов LTL-логики последовательностного и темпорального уровня, призванное обеспечить строгую однозначность интерпретации формул при реализации системы верификации на основе ассерций;

– создание формальной спецификации ключевых вычислительных процедур цикла работы последовательностных функций и темпоральных свойств в различных режимах интерпретации времени, ориентированных на достижение максимального быстродействия верификации.

2. Цикл работы ассерционного процесса в модели DRTLQ

Наиболее общим понятием модели DRTLQ является ассерционный процесс – совокупность элементов булевого, темпорального и верификационного уровня семантики ассерций, реализующая верификацию логически связанной группы темпоральных утверждений. Самым близким к ассерционному процессу языковым термином является единица верификации (verification unit) из языка PSL. Ассерционным процессом AP в модели DRTLQ называется целостная совокупность множества верификационных переменных B, списка текущих потоков активации Ω, очереди последовательностных функций F и темпоральных свойств P, одного ассерционного монитора M, как правило, относящаяся к определенному функциональному блоку системы и реализующая верификацию логически связанной группы темпоральных утверждений в процессе моделирования:

$$AP = \{B, \Omega, F, P, M\}. \quad (1)$$

В моменты изменения верификационные переменные создают события:

$$e = \{e_{\leftarrow}, e_{\uparrow\rightarrow}, e_{\downarrow\rightarrow}, \rho, t_b, t_a\}, \quad (2)$$

где $e_{\leftarrow}, e_{\uparrow\rightarrow}, e_{\downarrow\rightarrow}$ – связи события с другими событиями; ρ – характеристический вектор события; t_b и t_a – время создания и активации события соответственно, измеряемое в количестве тактовых событий с момента начала моделирования.

Событие называется активированным, если в ходе обработки оно достигло элемента ассерционного процесса, создающего поток активации. Поток (или кольцо) активации $\omega_k \in \Omega$ в модели DRTLQ – динамическое множество всех обрабатываемых событий e_k^i, $i \geq 0, k \geq 0$, соответствующих одному вычислительному пути π, который начинается с тактового цикла с порядковым номером k.

Событие может быть связано с другими событиями в двух различных видах цепочек. Правая цепочка $r_{\rightarrow}$, формируемая двунаправленными событийными связями $e_{\uparrow\rightarrow}, e_{\downarrow\rightarrow}$, соответствует событиям, связанным одним кольцом активации. Основная цель связывания событий в правые цепочки состоит в возможности каскадных операций над всем потоком (например, уничтожение при завершении потока). Левая однонаправленная цепочка $r_{\leftarrow}$, формируемая событийными связями $e_{\leftarrow}$, соответствует событиям различных колец активации, которые одновременно транспортируются через выбранный элемент модели. Правая цепочка неупорядочена, она формируется свободно, здесь порядок не играет никакой роли, а левая цепочка упорядочена по времени создания события, что принципиально важно для конвейерной обработки нескольких потоков активации.

Характеристический вектор события ρ – короткое двоичное слово, транспортируемое в составе события, индивидуальные биты которого моделируют семантические флаги-модификаторы, оказывающие влияние на обработку события. Важнейшие флаги-модификаторы приведены в табл.1.

Таблица 1

Обозначение	Описание
ρ^{VAL}	Текущее значение события
ρ^{LDEAD}	Событие должно быть отделено от левой цепочки
ρ^{RDEAD}	Событие должно быть отделено от правой цепочки
ρ^{ACTIVE}	Событие активировано
ρ^{RTOP}	Событие является вершиной потока активации
ρ^{ABORT}	Событие исходит от элемента прерывания анализа

Транспортирование событий осуществляется путем перегруппировки левых цепочек. Левые цепочки ассоциируются с элементами модели более высокого уровня, формирующими пути транспортирования событий. Пусть $r^1_{\leftarrow} : \{e^1_1 \dots e^1_{N_1}\}$ представляет собой левую цепочку, из которой транспортируются события, а $r^2_{\leftarrow} : \{e^2_1 \dots e^2_{N_2}\}$ – левую цепочку, к которой они направляются. Суть задачи простого (а) и копирующего транспортирования (б):

$$\begin{aligned} &\text{а)}\ r'^2_{\leftarrow} : r^2_{\leftarrow} \cup r^1_{\leftarrow}, \\ &\text{б)}\ r'^2_{\leftarrow} : \kappa(r^2_{\leftarrow}) \cup \kappa(r^1_{\leftarrow}) \end{aligned} \tag{3}$$

состоит в корректном расширении цепочки $r^2_{\leftarrow}$ событиями из цепочки $r^1_{\leftarrow}$ с сохранением относительного порядка событий в результирующей цепочке $r'^2_{\leftarrow}$:

$$\begin{cases} e_{\leftarrow}(e'^2_0) = \varepsilon; \\ \forall k, 1 \le k < \left| r'^2_{\leftarrow} \right| - 1, e'^2_{k-1} = e_{\leftarrow}(e'^2_k) \Rightarrow t_b(e'^2_{k-1}) < t_b(e'^2_k). \end{cases} \tag{4}$$

Слияние цепочек событий происходит начиная с правого края с одновременным итерированием в единственно допустимом направлении, при этом не теряется позиция вставки, что обеспечивает линейную сложность алгоритма относительно суммы $\left|r^1_{\leftarrow}\right| + \left|r^2_{\leftarrow}\right|$. Вставка предполагает изменение связей $e_{\leftarrow}$ непосредственно вносимого в цепочку события, а также события, следующего за ним. При копирующем транспортировании, оригинальные цепочки не модифицируются, в то время как простое транспортирование полностью очищает $r^1_{\leftarrow}$ и перезаписывает результатом $r'^2_{\leftarrow}$ цепочку $r^2_{\leftarrow}$.

Помимо основной задачи (3), осуществляется дополнительная задача элиминации лишних событий, руководствующаяся двумя критериями:

1. Если $\rho^{RDEAD}(e^k_1) = 1$, т.е. вычислительный поток события уничтожен, событие должно быть отключено от левой цепочки и уничтожено. Гарантируется, что событие с установленным флагом ρ^{RDEAD} будет уничтожено при первой же попытке транспортирования, соответственно, можно утверждать, что для любого события целевой цепочки $\rho^{LDEAD}(e^k_2) = 0$.
2. Если для двух соседних событий в результирующей цепочке выполняется условие $t_b(e'^2_{k-1}) = t_b(e'^2_k)$, а $\rho(e'^2_{k-1}) = \rho(e'^2_k)$ (характеристические векторы идентичны), то одно из событий можно уничтожить.

Частным случаем задачи транспортирования (3) является случай, при котором $|r^2_{\leftarrow}| = 0$. Такое упрощение сводит задачу к переносу тех элементов цепочки $r^1_{\leftarrow}$ в результирующую цепочку $r'^2_{\leftarrow}$, которые не попадают под действие описанных выше критериев элиминации. Этот процесс, аналогично полноценному транспортированию, характеризуется линейной сложностью, однако только относительно $\left|r^1_{\leftarrow}\right|$, и, в силу отсутствия необходимости редактирования событийных связей, имеет существенно лучшие показатели производительности.

Анализ ассерций в модели DRTLQ осуществляется по событию, доставка которого в ассерционный процесс является обязанностью родительской системы моделирования. Каждая итерация цикла работы всегда соответствует уникальному моменту времени. Ситуация, когда одному моменту времени соответствует более одной итерации анализа, исключается.

Итерации анализа разделяются на типы в зависимости от момента запуска:

– *регулярная* итерация – запускается по тактовому событию (если тактового события не определено, по событию изменения любой из множества верификационных переменных, принадлежащих ассерционному процессу);

– *итерация прерывания* – запускается при срабатывании условия прерывания (операторы *async_abort* и *sync_abort*), прерывает все активные вычислительные потоки путем обработки множеств внутренних событий элементов-очередей и элементов-репетиций;

– *финальная итерация* – запускается один раз в конце моделирования, транспортирует все цепочки событий непосредственно монитору на обработку с учетом требований условной допустимости (безусловная, слабая, сильная).

Финальная итерация и итерация прерывания предназначаются только для поддержки специальных видов ассерционных языковых конструкций и сводится к итерированию подготовленных заранее отдельных элементов системы (очередей и репетиций) и обработке множеств их внутренних событий.

Регулярная итерация анализа ассерций включает в себя следующие шаги:

1. Считывание значений всех ассерционных переменных.
2. Запуск шага вычислительной процедуры каждой из последовательностных функций, в соответствии с очередью обработки:

$$\begin{cases} e_{\leftarrow}(e'^{2}_{0}) = \varepsilon; \\ \forall k, 1 \le k < \left| r'_{\leftarrow} \right|^{2} - 1, e'^{2}_{k-1} = e_{\leftarrow}(e'^{2}_{k}) \Rightarrow t_b(e'^{2}_{k-1}) < t_b(e'^{2}_{k}). \end{cases} \quad (4a)$$

3. Запуск шага обработки ассерционного монитора.

Очередь обработки последовательностных функций упорядочивает запуск всех элементов таким образом, чтобы осуществить полную обработку модели за один проход. Первыми выполняются функции-генераторы, порождающие события от ассерционных переменных. Далее выполняются все последовательностные функции, в соответствии с иерархией.

Ассерционный монитор непосредственно не контактирует с уровнем последовательностей, а обрабатывает события по запросу. Для обеспечения взаимодействия уровня последовательностей с ассерционным монитором, вершиной иерархии последовательностных функций, т.е. последним элементом в очереди на выполнение, используется служебная функция действия f_{ACTION}. Данная функция не имеет ни внутренних, ни выходных событий, поскольку ее единственной задачей является доставка поступившей на вход цепочки событий к ассерционному монитору. Функции действия применяются также для реализации режима глобального времени, однако в этом случае цепочки поступивших событий доставляются не монитору, а темпоральным свойствам.

Пусть $r_{\leftarrow}$ – цепочка событий, транспортированных к монитору с последовательностного уровня. Независимо от типа монитора сначала проверяется статус активации каждого события: если $\rho^{ACTIVE}(e \in r_{\leftarrow}) \neq 1$, событие уничтожается без каких-либо нотификаций внешней среде. Дальнейшие шаги обработки событий на уровне монитора зависят от его типа.

Предположим, обрабатываемая ассерция моделирует инвариантное LTL-ограничение вида never({...seq...}) . В таком случае монитор называется запрещающим, и принцип его работы следующий:

$$\forall e \in r_{\leftrightarrow} \begin{cases} \rho^{VAL}(e) = 1 \Rightarrow \begin{cases} \text{report(failed);} \\ \text{recursively } \forall e' \in \{e_{\uparrow\rightarrow}, e_{\downarrow\rightarrow}\}(e), \rho^{RDEAD}(e') \leftarrow 1; \\ \rho^{LDEAD}(e) \leftarrow \rho^{RDEAD}(e) \leftarrow 1; \end{cases} \\ \rho^{VAL}(e) = 0 \Rightarrow e_{\uparrow\rightarrow}(e) = e_{\downarrow\rightarrow}(e) = e \Rightarrow \text{report(passed).} \end{cases} \quad (5)$$

Если значение события $\rho^{VAL}(e) = 1$, это означает, что дочерняя последовательность выполнена успешно, а значит, ассерция нарушена. В таком случае весь поток активации ω, соответствующий времени активации t_a рассматриваемого события $e \in r_{\leftarrow}$, признается удовлетворенным, и дальнейший его анализ (могут существовать другие события этого потока) не имеет смысла. Все события, относящиеся к данному потоку, уничтожаются моментально, используя связи события $e_{\uparrow\rightarrow}, e_{\downarrow\rightarrow}$ по правой цепочке. Монитор сигнализирует внешней среде о неудаче анализа ассерции, вызывая абстрактную процедуру *report* с неудачным статусом.

Если же значение события $\rho^{VAL}(e) = 0$, это означает, что один из сценариев последовательности удовлетворяет свойству. В таком случае, событие просто уничтожается из всех цепочек. Если событие было последним в собственном потоке активации, монитор сигнализирует внешней среде об успехе анализа ассерции, на этот раз вызывая процедуру *report* с положительным статусом.

Работа разрешающего монитора, моделирующего инвариантное ограничение вида always({...seq...}), подобна запрещающему монитору. Различие состоит только в противоположности условия срабатывания: событие с отрицательным значением $\rho^{VAL}(e) = 0$ является в данном случае определяющим неудачный статус и досрочное завершение анализа потока активации. Напротив, $\rho^{VAL}(e) = 1$ считается успешным результатом, и, если такое событие последнее в потоке активации, то ассерция удовлетворяется на данном вычислительном пути.

Также существует третий тип монитора – ожидающий, соответствующий LTL-ограничениям достижимости вида eventually!({...seq...}) :

$$\forall e \in r_{\leftarrow}, \begin{cases} \rho^{VAL}(e) = 1 \Rightarrow \begin{cases} \text{report(passed);} \\ \text{recursively} \\ \forall e' \in \{e_{\uparrow\rightarrow}, e_{\downarrow\rightarrow}\}(e), \rho^{RDEAD}(e') \leftarrow 1; \end{cases} \\ \rho^{VAL}(e) = 0 \Rightarrow \rho^{LDEAD}(e) \leftarrow \rho^{RDEAD}(e) \leftarrow 1. \end{cases} \quad (6)$$

В случае прихода события с положительным значением $\rho^{VAL}(e) = 1$ поток активации удовлетворяется, и монитор сигнализирует об успешном статусе анализа. Отрицательный статус $\rho^{VAL}(e) = 0$ лишь уничтожает поступившее событие. Никакие действия с потоком активации не происходят, поскольку в режиме локального времени свойство eventually! не может дать отрицательного результата, за исключением конца моделирования.

3. Семантика последовательностных функций

Последовательностной функцией $f \in F$ в модели DRTL Q называется некоторая вычислительная процедура, характеризующаяся множествами входных, внутренних и выходных событий, осуществляющая их пошаговое преобразование на каждом тактовом цикле верификации. Результатом последовательностной функции в некоторый момент времени (номер тактового цикла с начала моделирования) является содержимое множества выходных событий в этом моменте времени.

Активирующей последовательностной функцией является такая последовательностная функция, выходные события которой, во-первых, инициируют некоторый поток активации, соответствующий текущему тактовому циклу, если таковой еще не был инициирован, во-вторых, назначают связи между событиями по правой цепочке в соответствии с принадлежностью к инициированному потоку, в-третьих, устанавливают значение флага $\rho^{ACTIVE} = 1$.

С точки зрения обработки одного события последовательностные функции можно интерпретировать как конечные автоматы. Событие, поступив на вход последовательностной функции, попадает в некоторое начальное состояние. Из начального состояния в зависимости от особенностей вычислительной процедуры и внешних условий событие попадает в одно из внутренних состояний. Наконец, достигнув конечного состояния, событие передается на выход функции и используется следующими элементами модели.

Детали вычислительного процесса, скрытого последовательностной функцией, зависят от ее типа. Многие функции также предусматривают параметризацию. К основным последовательностным функциям относятся:

– функция-генератор – последовательностная функция, связанная с некоторой верификационной переменной $b \in B$, не имеющая входных и внутренних событий, мгновенно порождающая событие e в выходном множестве с текущим значением верификационной переменной:

$$f_{gen(b \in B)}(t) = [e_0 : \{e_0 \leftarrow e, \begin{matrix} e_{\uparrow\rightarrow} \\ e_{\downarrow\rightarrow} \end{matrix} = \varepsilon, \rho^{VAL} = b(t)\}]; \quad (7)$$

– функция конъюнктивного конкатенирования – последовательностная функция, имеющая два операнда, одним из которых обязательно является функция-генератор, значение выходных событий которого используется в качестве условия транспортирования цепочек событий из второго операнда:

$$f_{conj}(t, f_1, f_{gen}) : a = \rho^{VAL}(e^0 = first(f_{gen}(t))), \begin{cases} = 1 \Rightarrow f_{conj}(t, f_1, f_{gen}) = f_1(t); \\ a = 0 \Rightarrow f_{conj}(t, f_1, f_{gen}) = \varnothing; \end{cases} \quad (8)$$

где first(f (t)) – первое событие, принадлежащее множеству (в случае операнда-генератора первое событие является единственным);

– функции-очереди, имеющие единственный операнд и моделирующие временные интервалы;

– функции-репетиции, реализующие циклический анализ булевого выражения или другой последовательностной функции;

– функции, реализующие логические последовательностные операторы, такие как OR, AND, INTERSECT, WITHIN, а также импликации $|\rightarrow$ и $|\Rightarrow$.

Наиболее часто применяемой последовательностной функцией в предлагаемой модели является функция-очередь, лежащая в основе выбранного названия DRTLQ. Очереди применяются для моделирования интервалов между интересующими событиями в системе. Наглядно очередь можно представить в виде цепочки триггеров. На рис. 1 показана схема обработки последовательности событий {a;[*2]; b} на основе очереди, начинающейся с момента, в котором сигнал $a = 1$, и в прошествии 3 тактовых циклов завершающейся событием $b = 1$.

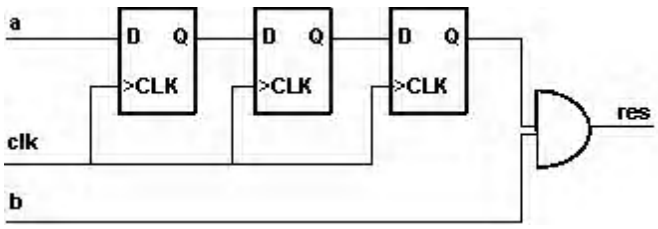

Рис. 1. Наглядная иллюстрация понятия DRTLQ-очередей

В данном случае функция-очередь $f_{\#}$ является операндом функции конъюнктивного конкатенирования f_{conj} с $f_{gen(b)}$ и имеет функцию-генератор $f_{gen(a)}$ на входе:

$$f(t, \{a;[*2]; b\}) = f_{conj}(t, f_{\#[3:3]}(t, f_{gen(a)}), f_{gen(b)}). \quad (9)$$

В общем виде, функция-очередь, имеющая идентичные конечные интервалы, функционирует путем хранения и продвижения к выходу списка цепочек событий, считанных с

входа в различные моменты моделирования, и упорядоченного по времени считывания. Цепочка событий, достигнувшая последней в очереди ячейки, является фактическим результатом функции на следующем такте обработки:

$$f_{\#[N:N]}(t, f_1(t)), N > 0 : \langle r_0(t), \ldots r_{N-1}(t) \rangle;$$
$$r_0(t) = f_1(t);$$
$$\forall i, 0 < i < N, r_i(t) = r_{i-1}(t-1);$$
$$f_{\#[N:N]}(t, f_1(t)) = r_{N-1}(t-1). \quad (10)$$

Задача усложняется, если значения интервалов очереди различаются. Копии событий, время нахождения которых в очереди превысило минимальный заданный интервал, но еще не достигло максимального интервала, направляются на выход. Сами события остаются внутри очереди до момента достижения максимального интервала и продолжают транспортирование по очереди. Копии событий связывают по правой цепочке:

$$f_{\#[N:M]}(t, f_1(t)), N > 0, N < M : \langle r_0(t), \ldots r_{M-1}(t) \rangle;$$
$$r_0(t) = f_1(t);$$
$$\forall i, 0 < i < M, r_i(t) = r_{i-1}(t-1);$$
$$f_{\#[N:M]}(t, f_1(t)) = r_{M-1}(t-1) \cup \bigcup_{i=N-1}^{M-2} \kappa(r_i(t-1)). \quad (11)$$

Если $M \to \infty$, суть вычислительной процедуры (11) не изменяется, однако структура внутренних цепочек событий $\langle r_0(t), \ldots r_{M-1}(t) \rangle$ преобразуется из статической формы в динамическую. Очередь с бесконечным интервалом будет генерировать и продвигать к выходу копии считанного события до тех пор, пока не будет остановлен породивший событие поток активации.

Если $N = 0$, копии считанных с входа событий моментально транспортируются к выходу, независимо от обработки события внутри очереди:

$$f_{\#[0:M]}(t, f(t)_1), M > 0 = f_{\#[1:M]}(t, f_1(t)) \cup \kappa(f_1(t)). \quad (12)$$

Функции-репетиции в модели DRTLQ разделяются на булевы и сложные, в зависимости от типа операнда (булево выражение и SERE соответственно). В целом, схема обработки булевых репетиций имеет определенное сходство с обработкой очередей. Дополнительно к сопоставлению времени регистрации в очереди, в репетициях проверяется значение входного булевого выражения. В том случае, если выражение не выполняется, все цепочки событий, в текущий момент хранящиеся в репетиции, поступают на выход со значением 0, и репетиция полностью очищается:

$$f_{[*N]}(t, f_{gen(b \in B)}(t), N > 0 : \langle v(t), r_0(t), \ldots r_{N-1}(t) \rangle;$$
$$v(t) = \rho^{VAL}(\text{first}(f_{gen(b)}(t)));$$

$$\begin{cases} r_0(t) = f_{gen(b)}(t); \\ v(t) = 1 \Rightarrow \forall i, 0 < i < N, r_i(t) = r_{i-1}(t-1); \\ f_{[*N]}(t, f_{gen(b)}(t)) = r_{N-1}(t); \\ \forall i, k, 0 \le i < N, 0 \le k < |r_i(t-1)|, \rho^{VAL}(e_k \in r_i(t-1)) \leftarrow 0; \\ v(t) = 0 \Rightarrow f_{[*N]}(t, f_{gen(b)}(t)) = \bigcup_{i=0}^{N-1} r_i(t-1); \\ \forall i, 0 \le i < N, r_i(t) = \varnothing. \end{cases} \quad (13)$$

Аналогично очереди (11), интервальная репетиция требует создания и продвижения к выходу копий цепочек событий, выдержанных внутри репетиции минимальное количество итераций, но не достигших максимума:

$$v(t)=1\Rightarrow f_{[*N:M]}(t,f_{gen(b\in B)}),\ N>0,\ N<M=\bigcup_{i=N}^{M-1}\kappa(r_i(t-1)). \tag{14}$$

Случаи $M\to\infty$ и $N=0$ определяются аналогичным очередям образом, разумеется, с учетом процедуры полного очищения репетиции при $v(t)=0$.

Реализация галопирующих репетиций модифицирует базовую репетиционную функцию (13) добавлением элемента $\varphi(t)$, который должен обеспечивать буферизацию последнего успешного входного события до момента появления следующего. Соответственно, галопирующая репетиция никогда не выполняет полную очистку, как обычная репетиция:

$$\begin{gathered}f_{[\to N]}(t,f_{gen(b\in B)}(t)),\ N>0:\langle v(t),\varphi(t),r_0(t),\llcorner r_{N-1}(t)\rangle,\\ v(t)=\rho^{VAL}(first(f_{gen(b)}(t)));\ \begin{cases}v(t)=1\Rightarrow\varphi(t)=f_{gen(b)}(t);\\ v(t)=0\Rightarrow\varphi(t)=\kappa(\varphi(t-1));\end{cases}\\ \begin{cases}r_0(t)=\varphi(t);\\ \forall i,0<i<N,\ r_i(t)=r_{i-1}(t-1);\\ f_{[\to N]}(t,f_{gen(b)})=r_{N-1}(t).\end{cases}\end{gathered} \tag{15}$$

Неконсекутивные репетиции расширяют реализацию галопирующих репетиций дополнительным выходным буфером, порождающим копии ранее транспортированных на выход событий до удовлетворения потока активации:

$$f_{[=N]}(t,f_{gen(b)}(t))=\kappa(f_{[\to N]}(t-1,f_{gen(b)}(t-1)))\cup f_{[\to N]}(t,f_{gen(b)}). \tag{16}$$

Наиболее сложными являются функции, реализующие логические операторы над последовательностями. Пусть имеется SERE $s=\{\{a;b\}\parallel\{c;d\}\}\&\&\{!e[*2]\}$. На рис. 2 в графическом виде представлена DRTLQ-реализация данного SERE, а формула (17) демонстрирует то же SERE в аналитическом виде.

$$\begin{gathered}f_s(t)=f^1_{\supset}(t);\ f^1_{\supset}(t)=join_{2-\&\&}(t,f^2_{\supset}(t),f_{[*2:2]}(t,f_{gen(\neg e)}(t)));\\ f^2_{\supset}(t)=join_{2-\parallel}\begin{pmatrix}t,\\ f_{conj}(t,f_{\#[1:1]}(t,f_{conj}(t,f^2_{\subset}(t),a)),b),\\ f_{conj}(t,f_{\#[1:1]}(t,f_{conj}(t,f^2_{\subset}(t),c)),d);\end{pmatrix}\\ f^2_{\subset}(t)=split_{2-\parallel}(t,f^1_{\subset}(t));\quad f^1_{\subset}(t)=split_{2-\&\&}(t,f_{gen(1)}(t)).\end{gathered} \tag{17}$$

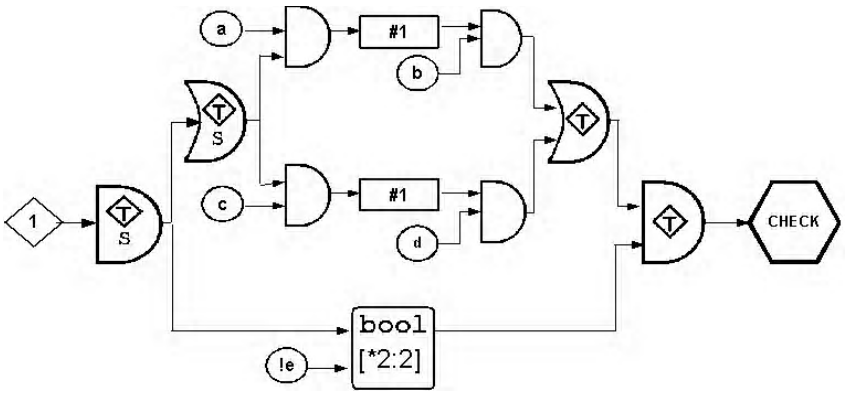

Рис. 2. DRTLQ-реализация последовательности с логическими операторами

Любой логический последовательностный оператор реализуется двумя DRTLQ-функциями: разделителем $f_{\subset}$ (splitter) и соединителем $f_{\supset}$ (joiner). Функция-соединитель является вершиной пути операндов, а начало каждого из операндов исходит из функции-разделителя. Ряд операторов (WITHIN, импликации) могут иметь только два операнда, но степень разветвления логических операций (AND,OR,INTERSECT) не ограничивается. Разделитель всегда имеет единственный вход и множество выходов, соответствующее количеству операндов. Событие, считываемое разделителем с входа, расслаивается на самостоятельные копии, каждая из которых транспортируется по пути одного из операндов. Соединитель имеет единственный выход, порождая выходные цепочки событий в соответствии с реализуемой функцией.

Аналогичным образом реализуются репетиции с операндами-последовательностями: вводятся функция-разделитель и функция-соединитель, причем разделитель, определяющий направление транспортирования событий, находится выше по иерархии, чем соединитель, помещаемый на нижнем уровне. В зависимости от количества итераций, пройденных событием в рамках репетиции, событие либо направляется разделителем на выход, либо возвращается к соединителю на очередную итерацию.

4. Семантика сложных темпоральных свойств

Темпоральное свойство $p \in P$ в модели DRTLQ – совокупность логических и темпоральных отношений между булевыми выражениями, последовательностными функциями и другими темпоральными свойствами, задающая пространство поведения системы. При помощи темпоральных свойств задаются утверждения о поведении системы на протяжении всего цикла моделирования. Элементы-свойства создаются только в режиме глобального времени и являются промежуточным звеном между последовательностными функциями и ассерционным монитором.

При наличии в модели уровня темпоральных свойств задачи проверки критериев удовлетворения потоков активации выполняются именно этими элементами. Вспомогательные последовательностные функции f_{ACTION} доставляют выходные цепочки событий на первый уровень иерархии темпоральных свойств. С ассерционным монитором взаимодействует только вершина иерархии свойств. Роль самого монитора упрощается и сводится к обработке статуса единственного за весь сеанс моделирования события.

Каждое свойство состоит из следующих структурных элементов:

$$p \in P : \{p_{\uparrow,} P_{\downarrow} \in P\} \times \{\mathrm{INIT}, \mathrm{ABORT}, \mathrm{EVAL}, \mathrm{FINAL}\}, \qquad (18)$$

где $p_{\uparrow,}$ – родительское свойство; $P_{\downarrow}$ – множество дочерних свойств; INIT – рекурсивная процедура активации вычислительного потока; ABORT – рекурсивная процедура отмены активации всех текущих вычислительных потоков; EVAL – процедура обработки события; FINAL – процедура обработки завершения моделирования.

Тело процедур и дополнительные элементы структуры различаются в зависимости от типа свойства. К основным типам свойств относятся:

– унарные свойства, в свою очередь подразделяющиеся на группы:

– буферные свойства (BUF, NOT);

– глобальные свойства (always , never , eventually!);

– бинарные свойства $(\rightarrow, \leftrightarrow, |\rightarrow, |\Rightarrow, \mathrm{until}*, \mathrm{before}*, \mathrm{next}*)$ – свойства, имеющие по два операнда-свойства;

– коммутативные свойства (& & , ||) – свойства, которые могут иметь неограниченное число равноправных операндов-свойств.

Унарные свойства в качестве операнда могут иметь как другие свойства, так и последовательностную функцию.

Цикл работы ассерционного процесса в режиме глобального времени отличается от простого режима дополнительными шагами по обработке уровня темпоральных свойств:

1. На каждом шаге моделирования осуществляется активация свойств путем вызова процедуры INIT на свойстве верхнего уровня. Активации обеспечивают разрешение создания вычислительных потоков на уровне последовательностей. В отличие от мониторов, всегда разрешающих активацию вычислительных потоков, свойство может блокировать часть дочерних последовательностей в случае необходимости.

2. События, транспортируемые функциями действия f_{ACTION} с последовательностного уровня, приводят к запуску процедуры EVAL у свойств первого уровня. В зависимости от этой процедуры, событие может быть либо уничтожено, либо транспортировано к родительскому свойству. Если событие признается удовлетворяющим процедуру EVAL свойства верхнего уровня, событие направляется к монитору, и весь цикл анализа ассерции завершается.

3. Финальная итерация предполагает вызов процедуры FINAL для всех свойств p, принадлежащих модели. Если анализ ассерции не был завершен на момент остановки симуляции, результат определяется именно в этот момент.

Рассмотрим простейшее темпоральное свойство BUF, которое либо может иметь последовательностный вход, создаваемый при продвижении булевого выражения или последовательности на уровень свойств, либо дочернее свойство-операнд:

$$p_{BUF}:\{p_{\uparrow}, \begin{cases} P_{\downarrow}=\varnothing, f_{ACTION}\in F; \\ P_{\downarrow}:\{p_1\in P\}; \end{cases} \begin{cases} \\ \end{cases}, \Omega_{BUF}\in\Omega\}\times \{INIT_{BUF}, ABORT_{BUF}, EVAL_{BUF}, FINAL_{BUF}\}, \quad (19)$$

где f_{ACTION} – дочерняя последовательностная функция; p_1 – дочернее свойство-операнд; Ω_{BUF} – множество активных вычислительных потоков, порожденных свойством. Очевидно, такое свойство не имеет дочерних свойств.

Процедура $INIT_{BUF}$ состоит в расширении Ω_{BUF} новым потоком активации ω_t, соответствующим текущему моменту времени, в который добавляется искусственное событие e^0_t. На уровне последовательностей наличие потока разрешает активацию событий, примыкающих по правой цепочке к e^0_t.

Процедура $ABORT_{BUF}$ состоит в очищении множества потоков Ω_{BUF} вместе с уничтожением всех порожденных правых цепочек событий.

Процедура $EVAL_{BUF}$, вызываемая либо функцией действия f_{ACTION} с последовательностного уровня, либо дочерним свойством p_1, удаляет из множества Ω_{BUF} поток ω', для которого время активации равно времени активации пришедшего с входа события $t_A(first(\omega')) = t_A(e)$. Само событие *e* направляется в процедуру EVAL родительского свойства.

Процедура $FINAL_{BUF}$ для всех вычислительных потоков Ω_{BUF} выбирает порожденное при активации событие e^0 и направляет к родительскому свойству: статус определяется режимом условной допустимости.

Аналогичным образом функционирует свойство NOT, имеющее идентичную структуру, а также процедуры активации и отмены активации:

$$p_{NOT}:\{p_{\uparrow}, \begin{cases} P_{\downarrow}=\varnothing, f_{ACTION}\in F; \\ P_{\downarrow}:\{p_1\in P\}; \end{cases} \begin{cases} \\ \end{cases}, \Omega_{NOT}\in\Omega\}\times \begin{cases} INIT_{NOT}=INIT_{BUF}; \\ ABORT_{NOT}=ABORT_{BUF}; \\ EVAL_{NOT}; \\ FINAL_{NOT}; \end{cases} \quad (20)$$

Отличие процедур $EVAL_{NOT}$ и $FINAL_{NOT}$ от аналогов в BUF состоит в инверсии значения $\rho^{VAL}(e)$, транспортируемого к родительскому свойству события.

Рассмотрим работу глобальных темпоральных свойств на примере свойства *always* :

$$p_G:\{p_{\uparrow,\downarrow}:\{P_\downarrow=\varnothing, f_{ACTION}\in F;\ \Omega_G\in\Omega\}\times\begin{cases}INIT_G;\\ ABORT_G=ABORT_{BUF};\\ EVAL_G;\\ FINAL_G;\end{cases}\quad P_\downarrow:\{p_1\in P\};\qquad (21)$$

Отличие процедуры $INIT_G$ от $INIT_{BUF}$ состоит в том, что в случае наличия дочернего свойства-операнда и открытых потоков активации в нем на каждом такте анализа создается дополнительный поток активации, помещаемый в Ω_{BUF} . Такой прием обеспечивает неявное ветвление потоков активации, порождаемых глобальными темпоральными операторами, без фактического ветвления вычислений на уровне дочерних последовательностей.

Процедура $EVAL_G$ состоит в завершении всех вычислительных потоков $\omega\in\Omega_G$, для которых выполняется соотношение $t_A(first(\omega))\le t_A(e)$, и транспортировании соответствующих потокам событий e^0 на уровень выше. Здесь ветвление свойств-операндов воссоздается, однако это не имеет такого эффекта на производительность анализа, как если бы производить все вычисления на последовательностном уровне.

Процедура $FINAL_G$ с успешным статусом завершает все стартовавшие потоки Ω_G. Очевидно, активность свойства *always* в момент остановки симуляции свидетельствует об успешном выполнении данного свойства.

Свойство *never* отличается от свойства *always* инверсией значений $\rho^{VAL}(e)$ транспортируемых событий в процедуре EVAL , а свойство *eventually*! – инверсией значений в процедуре FINAL.

Коммутативные свойства, такие как AND, имеют следующую структуру:

$$p_\wedge:\{p_{\uparrow,\downarrow}P_\downarrow:\{p_1,\ldots,p_N\},\mu(t):\langle k\in[0;N],v\in[0;1]\rangle\}\times\begin{cases}INIT_\wedge:\forall i,0\le i<N,INIT(p_i);\\ ABORT_\wedge:\forall i,0\le i<N,ABORT(p_i)\\ EVAL_\wedge;\\ FINAL_\wedge;\end{cases}\qquad (22)$$

где N – число операндов; $p_1,\ldots,p_N$ – свойства-операнды, а $\mu(t)$ – функция истории выполнения потока активации, каждый элемент которой соотносится с некоторым тактовым циклом и включает информацию о количестве завершенных на текущий момент операндов, а также общий статус завершения.

Работа процедур $INIT_\wedge$ и $ABORT_\wedge$ тривиальна и состоит из вызова аналогичных процедур на всех свойствах-операндах по очереди.

Работа процедуры $EVAL_\wedge$ состоит из двух шагов:

1. Изменение статуса истории потока активации по принципу $v(\mu(t))\leftarrow v(\mu(t))\wedge\rho^{VAL}(e)$ (очевидно, начальное значение $v=1$). В случае изменения статуса $v(\mu(t-1))=1\rightarrow v(\mu(t))=0$ поток активации признается неудачным, и событие e продвигается на уровень выше.

2. Инкремент числа выполненных операндов: $k(\mu(t))\leftarrow k(\mu(t))+1$. Если новое значение $k(\mu(t))=N$, поток активации признается успешным, и событие e также продвигается на уровень выше.

Работа процедуры $FINAL_\wedge$ заключается в продвижении на уровень выше событий всех потоков активации, для которых имеется хотя бы один незавершенный поток операнда: $k(\mu(t))<N$ – а начальный статус по-прежнему не изменился: $v(\mu(t))=1$.

Отличие свойства OR в деталях вычислительных процедур. Во-первых, начальным значением $v(\mu(t))$ является 0. Во-вторых, условие досрочного завершения анализа потока в процедуре $EVAL_{\vee}$ заключается в переходе $v(\mu(t-1)) = 0 \to v(\mu(t)) = 1$, и продвигает успешный статус. В-третьих, достижение значения $k(\mu(t)) = N$, при котором $v(\mu(t)) = 0$, завершает поток с неудачным статусом.

Аналогичным способом строится обработка темпоральных свойств $\to$ и $\leftrightarrow$, с той разницей, что число операндов здесь всегда равно двум.

Бинарное свойство, реализующее семейство операторов *until* *, имеет более сложную структуру и вычислительные процедуры:

$$p_U : \{p_\uparrow, P_\downarrow : \{p_1, p_2\}, v(t) :\in \{0,1,X\}, \{s_1, s_2, s_1', s_2'\}(t) :\in \{0,1,X\}\} \times \{INIT_U, ABORT_U = \{ABORT(p_1); ABORT(p_2)\}, EVAL_U, FINAL_U\}; \quad (23)$$

где p_1, p_2 – дочерние свойства-операнды; $v(t)$ – функция текущего статуса вычислительного потока; $s_1(t)$ и $s_2(t)$ – функции фактического состояния потоков, принадлежащих операндам, а $s_1'(t)$ и $s_2'(t)$ – функции интерпретации состояния потоков, принадлежащих операндам.

Процедура $INIT_U$ активирует свойства-операнды и кроме того создает записи для текущего момента *t* с начальными значениями $s_1(t) = s_2(t) = X$, а также $v(t) = X$.

Работа процедуры $EVAL_U$ состоит из следующих шагов:

1. Обновление состояния операнда, с которого пришло событие: $s_{i1,2}(t) \leftarrow \rho^{VAL}(e)$.

2. Обновление состояния ранее не разрешенных активированных потоков, начиная со стартовавших одновременно с событием, до текущего времени:

$\exists k, 0 \le k < (t_{now} - t_A(e))$,

$$s2_2(t_A(e)+k) = \begin{cases} 0 \Rightarrow \begin{cases} s_1'(t_A(e)+k) = 0 \Rightarrow v(t_A(e)+k); \\ v(t_A(e)+k) \leftarrow X; \end{cases} \\ 1 \Rightarrow \begin{cases} k > 0 \Rightarrow v(t_A(e)+k) \leftarrow s_1'(t_A(e)+k-1); \\ k = 0 \Rightarrow v(t_A(e)+k) \leftarrow 1; \end{cases} \\ X \Rightarrow v(t_A(e)+k) \leftarrow X; \end{cases} \quad (24)$$

3. Разрешение потоков (отправка события на уровень выше), для которых $v(t)$ на предыдущем шаге претерпело изменение $X \to 0$ или $X \to 1$.

В табл. 2 приведены определения функций $s_1'(t)$ и $s_2'(t)$ на основе функций $s_1(t)$ и $s_2(t)$, изменяющиеся в зависимости от конкретного оператора.

Таблица 2

Оператор	$s_1'(t)$	$s_2'(t)$
f_1 until f_2	$s_1(t)$	$s_2(t)$
f_1 until_ f_2	$s_1(t)$	$s_1(t) \wedge s_2(t)$
f_1 before f_2	$\neg s_2(t)$	$s_1(t) \wedge \neg s_2(t)$
f_1 before_ f_2	$\neg s_2(t)$	$s_1(t) \wedge \neg s_2(t)$

Процедура $FINAL_U$ для всех потоков с $v(t) = X$ направляет событие, принадлежащее потоку, на родительский уровень со статусом, определяющимся соотношением $\neg strong(p_U) \wedge s_1(t_\infty)$ для операторов $before^*$ и $\neg strong(p_U) \wedge s_2(t_\infty)$ – для операторов $until^*$.

5. Выводы

Задача функциональной верификации параллельных систем состоит в проверке логико-временных ограничений между атомарными утверждениями, ассоциированными с состояниями системы на всех возможных вычислительных путях. Утверждения об удовлетворении системой определенных ограничений, называемые ассерциями, отражают требования спецификации в виде формальной математической формулы, использующей аппарат темпоральной логики, который пришел в компьютерную инженерию из философии. Экспоненциальная сложность формальной проверки накладывает существенные ограничения на применимость метода к крупным моделям. Лучшую производительность верификации LTL-ограничений обеспечивают динамические методы, функционирующие путем конвертирования LTL к детерминированному конечному автомату. Несмотря на проблему низкого покрытия пространства состояний системы (зависящую от внешнего теста) в динамических методах, а также нелинейный рост размера проверяющих автоматов в зависимости от структурной сложности формулы, использование LTL-логики во время симуляции способно обеспечить обнаружение большинства функциональных нарушений с более приемлемой производительностью по сравнению с формальными методами.

Для обеспечения существенно лучшей производительности анализа ассерций и поддержки режима полноценной интерпретации глобальных темпоральных операторов была предложена модель динамических регистровых очередей (DRTLQ). Она ориентирована на высокопроизводительный анализ элементов линейной темпоральной логики и эффективность обнаружения и локализации нарушений. Модель состоит из четырех структурных уровней: 1) переменных верификации, обеспечивающих абстракцию выражений булевого уровня, 2) последовательностных функций, предназначенных для реализации регулярных выражений, учитывающих параметр времени, 3) темпоральных свойств и ассерционного монитора, контролирующих потоки активации при анализе формул, 4) ассерционного процесса, реализующего верификацию логически связанной группы темпоральных утверждений. В отличие от модели на основе конечных автоматов, количество элементов модели растет линейно относительно размера моделируемой LTL-формулы. Высокий уровень компактности модели достигается за счет минимальной структурной сложности представления простых и интервальных временных сдвигов, циклических ограничений, логических разветвлений – конструкций, порождающих сложные деревья вычислительных процессов, приводящих модель на основе конечных автоматов к нелинейному росту количества состояний.

Событийная модель обработки конструкций-свойств и вычислительные процедуры, контролирующие множества инициированных вычислительных путей, обеспечивают эффективную реализацию всех LTL-операторов с линейной сложностью. Производительность достигается иерархической декомпозицией событийных потоков. Теоретическая древовидность низкоуровневых вычислений, определяемая семантикой LTL, подменяется линейной нагрузкой на последовательностный уровень, а фактический учет сложных ветвлений эффективно реализуется на высоком уровне абстракции. Вычислительная сложность реализации такова, что применение на практике оказывается существенно быстрее (от 25 до 500%) по сравнению с традиционным способом реализации на основе конечных автоматов.

Ключевым *научным результатом* работы является формальное определение семантики сложных операторов LTL-логики последовательностного и темпорального уровня, а

также создание формальной спецификации ключевых вычислительных процедур цикла работы последовательностных функций и темпоральных свойств в различных режимах интерпретации времени.

Основным *практическим результатом* исследования является разработка модели процесса реализации сложных операторов линейной темпоральной логики, характеризующаяся линейной вычислительной сложностью относительно обработки одного вычислительного пути, возможностью одновременной обработки нескольких вычислительных путей за счет конвейерной архитектуры. Предложенные алгоритмы обеспечивают системе анализа ассерций параметры быстродействия, позволяющие уверенно конкурировать с аналогичными системами на EDA-рынке.

Список литературы: 1. *Emerson E.A.* Temporal and modal logic // In J. Van Leeuwen, editor, "Handbook of Theoretical Computer Science", Elsevier Science Publishers, 1990, volume B. P. 996–1072. **2.** *Clarke E.M.Jr, Grubmerg O., Peled D.A.* Model Checking // The MIT Process, 2002. 314p. **3.** *Drechsler R.* Advanced Formal Verification // Kluwer Academic Publishers, 2004. 277p. **4.** *Foster H., Krolnik A., Lacey D.* Assertions-based Design", Kluwer Academic Publishers, 2003. 392p. **5.** *Forczek M, Hrynkiewicz K.* Formal properties evaluation // Proc. of Programmable Devices and Systems Workshop 2003 (PDS-2003). P.305-311. **6.** *Хаханов В.И., Зайченко С.А.* Модель динамических регистровых очередей для быстродействующего анализа линейных темпоральных ограничений // АСУ и приборы автоматики. 2007. Вып. 136. С. 10-25. **7.** *Хаханов В. И., Егоров А. А., Зайченко С. А., Обризан В. И., Каминская М. А.* Механизм ассерций для функциональной верификации проектируемых цифровых систем // АСУ и приборы автоматики. 2005. Вып.130.С. 85-95. **8.** *Зайченко С.А., Хаханов В.И.* Эффективная функциональная верификация моделей цифровых систем на кристалле на основе ассерций глобального времени // АСУ и приборы автоматики. 2007. Вып. 137.С.4-13.

Поступила в редколлегию 19.09.2008

Зайченко Сергей Александрович, аспирант кафедры АПВТ ХНУРЭ, начальник отдела разработки компании Aldec-Kharkov Ltd. Научные интересы: системы автоматизированного проектирования, моделирования и верификации цифровых систем на кристаллах. Увлечения: литература, музыка, футбол. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (097)-367-62-93. E-mail: Sergei.Zaychenko@aldec.com

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (057)-702-13-26. E-mail: hahanov@kture.kharkov.ua

РЕФЕРАТИ

УДК 681.326:519.613

Формальна семантика складних операторів лінійної темпоральної логіки / С.О. Зайченко, В.І. Хаханов // АСУ та прилади автоматики. 2008. Вип. 145. С.14-28.

Розглянута проблема відсутності чіткості інтерпретації складних операторів лінійної темпоральної логіки при застосуванні до аналізу цифрових систем як частини динамічних методів верифікації. Введені формальні визначення та специфікації ключових обчислювальних процедур робочого циклу верифікації, що орієнтуються на максимальну швидкодію аналізу.

Табл. 2. Іл. 2. Бібліогр.: 8 назв.

UDC 681.326:519.613

Formal semantics of complex linear temporal logic operators / S. O. Zaychenko, V.I. Hahanov // Management Information System and Devises. 2008. N 145. P.14-28.

This paper considers a problem of interpretation ambiguity for complex linear temporal logic operators applied to the verification of digital systems within dynamic verification methods. Paper introduces formal definitions and specifications for key computational procedures of assertion verification cycle, which target maximum possible analysis performance.

Tab. 2. Fig. 2. Ref.: 8 items.