УДК 681.326:519.613

С.А. ЗАЙЧЕНКО, С.В. ЧУМАЧЕНКО

ОПТИМИЗАЦИИ ВЫЧИСЛИТЕЛЬНОГО ЦИКЛА АНАЛИЗА АССЕРЦИЙ

Предлагается обзор наиболее значимых оптимизаций вычислительного цикла модели DRTLQ, ориентированных на уменьшение количества элементарных действий по транспортированию событий от входов к выходам модели. Разрабатывается метод компрессии событий на выходе бесконечных репетиций.

Введение. Исследование связано с созданием новых моделей и методов верификации, а также с эффективным устранением ошибок: 1) допускаемых инженерами в системной модели, тестах и спецификации в процессе проектирования; 2) обусловленных несовершенством средств диагностирования в системах автоматизации, затрудняющих локализацию и устранение причины возникновения ошибки; 3) обусловленных недостаточной производительностью и точностью программных систем автоматизации, качество которых растет существенно медленнее увеличения сложности обрабатываемых моделей. Комплексное решение проблемы верификации системных моделей позволит в значительной степени снизить затраты на проектирование цифровых систем на кристаллах. Согласно исследованиям ведущих мировых компаний в области EDA (Cadence Design Systems, Synopsys Inc., Mentor Graphics Corporation, Magma, IBM, Intel, Sun Microsystems, Cisco Systems Inc., Atrenta, Aldec Inc.) усилия ученых должны быть сосредоточены на создании эффективных методов верификации, способных: 1) в десятки раз снизить вероятность возникновения ошибок за счет уменьшения участия человека в процессе проектирования; 2) обеспечить обнаружение и диагностирование допущенных неточностей на ранней стадии проектирования в целях сокращения времени и стоимости устранения несоответствий спецификации; 3) на порядок повысить производительность и надежность систем верификации за счет повышения уровня абстракции как самих моделей, так и тестовых воздействий.

Предлагается обзор наиболее значимых оптимизаций вычислительного цикла модели DRTLQ, ориентированных на уменьшение количества элементарных действий по транспортированию событий от входов к выходам модели. Рассматриваются только высокоуровневые быстродействующие алгоритмические методы, а также минимальные модификации изложенных выше структур данных, обеспечивающие существенное ускорение цикла обработки темпоральных описаний. Следует отметить не менее важную роль низкоуровневых задач оптимизации вычислений на булевом уровне. Хотя модель DRTLQ [2] предполагает полное абстрагирование от деталей элементарных вычислений через абстракцию верификационных переменных, за этим уровнем модели скрыта реализация компилятивного типа, где применяются оптимизации ассемблерного уровня. Такие оптимизации подбираются для конкретной платформы выполнения, и носят аналогичный характер по

сравнению с оптимизациями, применяемыми для ускорения вычислений в процессе HDL-симуляции.

Цель исследования – разработка моделей и методов функциональной верификации цифровых систем на кристаллах на основе использования темпоральных ассерций для тестового диагностирования ошибок в процессе программно-аппаратного моделирования, обеспечивающего существенное повышение качества цифрового изделия, а также уменьшение временных и материальных затрат проектирования.

Задачи исследования: 1) аналитический обзор наиболее значимых оптимизаций вычислительного цикла модели DRTLQ, ориентированных на уменьшение количества элементарных действий по транспортированию событий от входов к выходам модели: 2) разработка метода компрессии событий на выходе бесконечных репетиций.

Источники: языки описания аппаратуры высокого уровня [3-5,16], моделирование и синтез цифровых систем на кристаллах [6, 7, 10, 11, 15], верификация цифровых проектов [8,9,12-14], ассерции [17-19].

1. Метод предикторных связей. При транспортировании событий через последовательностный уровень модели DRTLQ распространенной ситуацией может быть продвижение события с $\rho^{VAL} = 0$. Существует ряд конструкций, например, логические последовательностные операторы AND и INTERSECT, когда результат функции зависит от появления такого события, и оно является ожидаемым. Во многих же других случаях транспортирование такого события является признаком предстоящего неудачного разрешения вычислительного потока. Пусть имеется последовательность $\{a;b[*2];c\}$ (рис. 1). Любой из выделенных генераторов может выдать событие с $\rho^{VAL} = 0$ и однозначно определить результирующий негативный статус анализа всей последовательности.

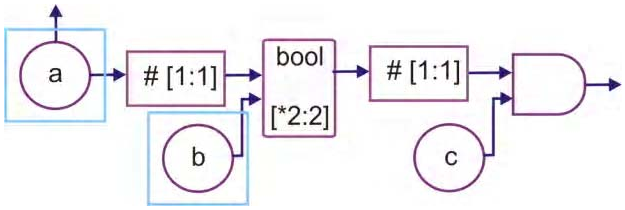

Рис. 1. Потенциальные точки досрочного появления неудачного статуса анализа последовательности $\{a;b[*2];c\}$

Учитывая значительное количество элементов, стоящих на пути формулы после первого и второго генераторов, имеет смысл отказаться от дальнейшего продвижения события с $\rho^{VAL} = 0$ и доставить неудачное событие непосредственно на следующий по иерархии уровень обработки. Введение быстрой прямой связи транспортирования события, однозначно досрочно определяющего неудачный статус анализа последовательности, к следующему уровню структур модели (ассерционный монитор или темпоральное свойство) называется методом предикторных связей (рис. 2).

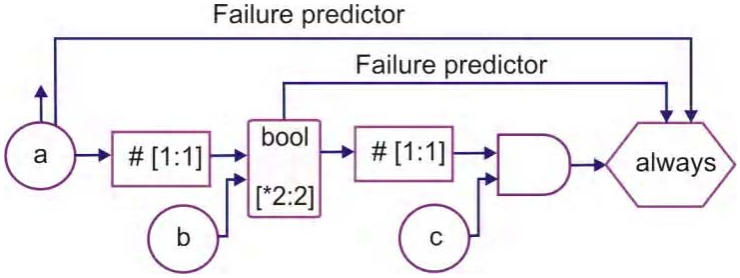

Рис. 2. Понятие предикторных связей

Предикторная связь к следующему уровню создается от элементов, непосредственно наблюдающих состояние, способное досрочно завершить анализ последовательности. В примере на рис. 2 первая предикторная связь создается от начальной функции-генератора (ситуация $a = 0$), а вторая – от элемента репетиции (ситуация $b = 0$). Вторая связь не

задается непосредственно на элементе-генераторе, поскольку само по себе появление события $b = 0$ не является определяющим фактором разрешения анализа, так как требуется выполнение $a = 1$ на предыдущем такте анализа. Последовательность может быть также разрешена с неудачным статусом событием $c = 0$, однако в данном случае добавление предикторных связей не имеет смысла, так как за последним элементом-генератором более нет элементов с временным сдвигом.

Выявление и доставка событий с $\rho^{VAL} = 0$ по предикторной связи обеспечивается модификацией процедуры левого итерирования цепочек в соответствующих контейнерах, описанной в [1]. Помимо анализа флага разрешения правой цепочки, альтернативным условием отделения события от цепочки транспортирования является неудачный статус анализа. В таком случае событие, вместо немедленного уничтожения, доставляется в точку назначения по предикторной связи. Те же событийные контейнеры, для которых предикторных связей не предусматривается, функционируют согласно собственным описанным выше процедурам без модификаций.

Реакция ассерционного монитора на появление событий с негативным статусом по предикторной связи зависит от его типа. Разрешающий монитор для ограничения вида $always(\{...seq...\})$ полностью уничтожает поток активации, поскольку ассерция однозначно считается нарушенной. Запрещающий монитор вида $never(\{...seq...\})$ при получении события $\rho^{VAL} = 0$, напротив, уничтожает его, и лишь в том случае, когда событие является единственным (последним) в потоке активации; это приводит к разрешению последовательности, причем с положительным статусом. К ожидающим мониторам вида $eventually!(\{...seq...\})$ также формируются предикторные связи, роль которых состоит только в элиминации неудачных витков анализа за счет досрочного уничтожения замеченного неудачного события.

Отключение предикторных связей не влияет на функциональную составляющую результата. Очевидно, если формула не удовлетворяется на некотором вычислительном пути, модель DRTLQ продемонстрирует этот факт независимо от наличия предикторных связей. Отличие состоит в конкретном моменте обнаружения неудачного результата (таб. 1). Количество вычислительных путей, завершившихся неудачно, либо успешно, либо не завершившихся вообще, не отличается между вариантами модели. Однако статус $FAIL^2$ в модели с учетом предикторов получен на первом же такте анализа ($a = 0$ – срабатывает первая предикторная связь), статусы $FAIL^4$ и $FAIL^5$ также получены досрочно ($b = 0$ – работа второй предикторной связи). Условие $c = 0$, а также успешный статус обнаруживаются одновременно.

Таблица 1. Ускорение анализа при помощи предикторных связей

#	A	b	c	Базовая модель	Предикторы
1	1	1	1	-	-
2	0	1	1	-	$FAIL^2$
3	1	1	1	-	-
4	1	1	1	$PASS^1$	$PASS^1$
5	1	1	1	$FAIL^2$	-
6	1	0	0	$FAIL^3$	$FAIL^3$ $FAIL^4$ $FAIL^5$
7	1	1	1	$FAIL^4$	-
8	1	1	1	$FAIL^5$	-
Конечный момент симуляции				$UNFINISHED^{6-8}$	$UNFINISHED^{6-8}$

Число операций транспортирования индивидуальных событий в тесте, рассмотренном в табл. 1, равно 31 в базовой модели, и лишь 20 – в модели с учетом предикторов. Кроме

того, за счет раннего завершения FAIL[2] по первой предикторной связи было создано/ уничтожено на одно меньше событийное расширение для обработки репетиции. Поскольку влияние времени проверки события на наличие признака $\rho^{VAL} = 0$ при итерировании контейнеров событий ничтожно мало, преимущества внесения предикторных связей очевидны с точки зрения производительности.

2. Метод обратного оповещения интервальных очередей и репетиций. Пусть имеется последовательность $\{a;[*3:5];b\}$, использующая интервальную форму элемента-очереди (рис. 3), на которую подается тестовое воздействие, представленное в табл. 2.

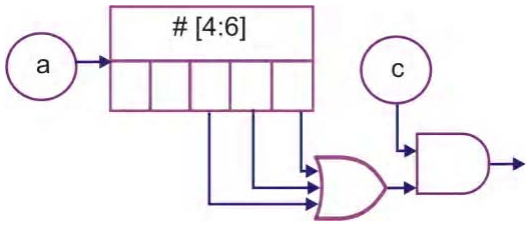

Рис. 3. Последовательность с интервальной формой очереди

Таблица 2. Тестовое воздействие на функцию-очередь

#	A	b	Содержимое очереди	Выход очереди	Результат
1	1	0	$\varnothing$	$\varnothing$	-
2	1	0	$[1]: e_1^1$	$\varnothing$	-
3	1	0	$[1]: e_1^2; [2]: e_1^1$	$\varnothing$	-
4	1	0	$[1]: e_1^3; [2]: e_1^2; [3]: e_1^1$	$\varnothing$	-
5	1	0	$[1]: e_1^4; [2]: e_1^3; [3]: e_1^2; [4]: e_1^1$	e_2^1	-
6	1	0	$[1]: e_1^5; [2]: e_1^4; [3]: e_1^3; [4]: e_1^2; [5]: e_1^1$	$e_3^1; e_2^2$	-
7	1	1	$[1]: e_1^6; [2]: e_1^5; [3]: e_1^4; [4]: e_1^3$ $[5]: e_1^2$ $[6]: e_1^1$	$e_4^1; e_3^2; e_2^3$	PASS[1]
8	1	1	$[1]: e_1^7; [2]: e_1^6; [3]: e_1^5; [4]: e_1^4; [5]: \varnothing; [6]: \varnothing$	e_2^4	PASS[2]

Столбец “Содержимое очереди” в табл. 2 демонстрирует динамику изменения множеств внутренних событийных контейнеров интервальной очереди, а столбец “Выход очереди” – множество событий, поступающих на выход очереди на каждом такте. В соответствии с семантикой интервальных очередей, определенной в [2], на выход очереди подаются копии событий из внутренних множеств, которые достигли необходимого минимального интервала ожидания, но еще не достигли максимально допустимого.

Следует отметить, что сгенерированные копии событий на тактовом цикле №5 (e_2^1) и №6 ($e_3^1; e_2^2$) оказываются невостребованными при достижении элемента конъюнктивной конкатенации, поскольку $b = 0$ на данных тактах. Варианты развития последовательности не удовлетворяются в таком случае, и сгенерированные копии событий нужно уничтожить. Если разница между границами интервала очереди велика, избыточно может генерироваться и через короткое время уничтожаться большое количество событий.

Аналогичная ситуация характерна и для элементов, реализующих булеву репетицию интервальной формы, где в соответствии с [2] также генерируются копии внутренних событий, превзошедших минимальное количество вычислительных итераций.

Избежать такого нерационального поведения позволяет метод обратного оповещения интервальных очередей и репетиций. Суть метода заключается в добавлении дополнитель-

ной управляющей обратной связи между интервальным последовательностным элементом и элементом-генератором, от которого он зависит (рис. 4).

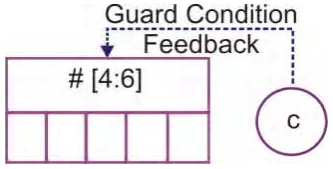

Рис. 4. Связь обратного оповещения интервальных очередей

Семантика функции-очереди модифицируется с учетом связи обратного оповещения – булевого значения GCF – таким образом, что ложное значение, возникающее на следующем элементе-генераторе, блокирует копирование лишних цепочек внутренних событий ввиду нецелесообразности их дальнейшего транспортирования:

$$f_{\#[N:M]}(t) = r_{M-1}(t-1) \cup \begin{cases} \bigcup_{i=N-1}^{M-2} \kappa(r_i(t-1)), GCF = 1; \\ \varnothing, GCF = 0. \end{cases} \quad (1)$$

При выполнении условия GCF = 0 на выход очереди поступают только те события, которые достигли максимального интервала ожидания в очереди. Такие события не будут уничтожены при прохождении элемента конъюнктивной конкатенации, однако им будет присвоен атрибут $\rho^{VAL} = 0$.

В табл. 3 представлены результаты моделирования теста, аналогичного тесту из табл. 2, с учетом связи обратного оповещения.

Таблица 3. Работа очереди с учетом оптимизации GCF

a	b (условие GCF)	Содержимое очереди	Выход очереди	Результат
1	0	$\varnothing$	$\varnothing$	-
1	0	[1]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^2 ; [2]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^3 ; [2]: e_1^2 ; [3]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^4 ; [2]: e_1^3 ; [3]: e_1^2 ; [4]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^5 ; [2]: e_1^4 ; [3]: e_1^3 ; [4]: e_1^2 ; [5]: e_1^1	$\varnothing$	-
1	1	[1]: e_1^6 ; [2]: e_1^5 ; [3]: e_1^4 ; [4]: e_1^3 ; [5]: e_1^2 [6]: e_1^1	e_4^1 ; e_3^2 ; e_2^3	PASS[1]
1	1	[1]: e_1^7 ; [2]: e_1^6 ; [3]: e_1^5 ; [4]: e_1^4 ; [5]: $\varnothing$ [6]: $\varnothing$	e_2^4	PASS[2]

3. Метод иерархического разрешения логических групп. Пусть имеется сложная LTL-формула, содержащая иерархически вложенные логические последовательностные операторы OR и AND: $(\{a;b[*2]\} | \{a;[*2];b\}) | (\{c;[*2];d\} \& \{e;d;c\})$. На рис. 5. представлена статическая структура реализации данной формулы в модели DRTLQ. Для удобства последующего анализа элементы модели пронумерованы.

Пара элементов OR 1/25 образует первый уровень логического разветвления, а пары элементов OR 2/11 и элементов AND 12/24 – второй уровень. Каждый из уровней прикрепляет к поступающим на вход событиям соответствующие реализуемым операторам событийные расширения. События, проходящие через элемент 9, также дополняются третьим расширением, необходимым для обработки репетиции. Единственное поступившее на вход

разделителя 1 событие породит 3 копии, 4 событийных расширения, и в сумме потребует максимум 20 операций транспортирования в течение 4 циклов анализа. Из-за пересечения параллельно анализируемых последовательностей в модели одновременно может существовать до 4 потоков активации, в общей сложности включающих 16 событий и 15 расширений. Получение результата первого вычислительного пути может потребовать выполнения около 50 операций транспортирования событий.

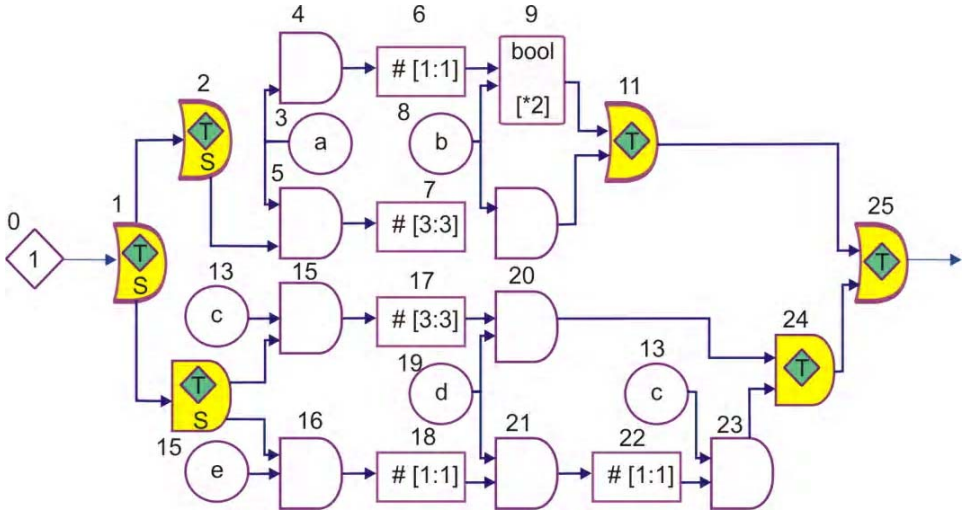

Рис. 5. Модель DRTLQ с вложенными последовательностными операторами

Представим сценарий возможного теста, при котором сигнал $a = 0$ и одновременно с этим либо $c = 0$, либо $e = 0$. В таком случае вычислительный путь, начинающийся с такого вектора, может быть удовлетворен на первом же такте анализа, поскольку на всех возможных вариантах ветвления модели будут транспортироваться события с $\rho^{VAL} = 0$. Однако моментальное разрешение потока активации является проблематичным в модели на рис. 5, поскольку здесь в силу структуры модели нельзя применить метод предикторных связей. Без модификации рабочего цикла модели такие события придется транспортировать к выходам за несколько циклов анализа, несмотря на то, что результат вычисления в целом уже однозначно определен на первом же шаге.

Решением данной проблемы является предлагаемый метод иерархического разрешения логических групп. Метод основывается, во-первых, на дополнении логических событийных расширений [1, формула (3.5)] связями быстрого транспортирования к элементу-соединителю, во-вторых, на учете атрибута $\rho^{RESOLVED}$ расширения события при итерировании событийного контейнера. Основная идея метода состоит в отказе от транспортирования событий уже разрешенной логической группы, а также на быстром продвижении разрешающего события к выходам модели. Подробно рассмотрим предлагаемый алгоритм на примерах:

1)В начальном состоянии все событийные контейнеры модели пусты. Элементом 0 генерируется первое событие e_1^1, создающее поток активации. Событие транспортируется к элементу-разделителю 1, где создается копия e_2^1, и к обоим событиям прикрепляется расширение X_L^1, инициализируемое значениями

$$< \rho^{VAL} = 1, \rho^{RESOLVED} = 0, N_T = 2, N_C = 0 >,$$

а также содержащее связь быстрого транспортирования к элементу 25. Далее события следуют к элементам 2 и 12 соответственно, где создаются дополнительные копии события e_3^1 и e_4^1, а также прикрепляются вложенные расширения X_L^2 со связью с элементом 11 (для события, транспортируемого по ветке с OR-разделителем 2) и X_L^3 со связью с элементом 24 (для события, транспортируемого по ветке с AND-разделителем 12). В табл. 4 показано текущее состояние обработки.

2) Производится анализ элемента 4, изменяющего атрибут $\rho^{VAL}(e_1^1)$ с 1 на 0 в связи с ложным значением, считываемым с генератора 3 при $a=0$. Транспортировать данное событие обычным путем далее не имеет смысла, и по быстрой связи оно направляется к элементу 11. Элемент не разрешает группу, но увеличивает число N_c и уничтожает событие e_1^1 (см. табл. 5).

Таблица 4. Состояние обработки ассерции на шаге №1

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^1	VAL = 1 RDEAD = 0 LDEAD = 0	4-in	e_1^1	e_1^4 - e_1^2	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^2	VAL = 1 RDEAD = 0 LDEAD = 0	5-in	e_1^2	e_1^1 - e_1^3	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^3	VAL = 1 RDEAD = 0 LDEAD = 0	15-in	e_1^3	e_1^2 - e_1^4	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^4	VAL = 1 RDEAD = 0 LDEAD = 0	16-in	e_1^4	e_1^3 - e_1^1	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25

Таблица 5. Состояние обработки ассерции на шаге №2

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^1	VAL = 0 RDEAD = 1 LDEAD = 1	11-in	$\varnothing$	$\varnothing$	OR2 NC=1 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^2	VAL = 1 RDEAD = 0 LDEAD = 0	5-in	e_1^2	e_1^4 - e_1^3	OR2 NC=1 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^3	VAL = 1 RDEAD = 0 LDEAD = 0	15-in	e_1^3	e_1^2 - e_1^4	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^4	VAL = 1 RDEAD = 0 LDEAD = 0	16-in	e_1^4	e_1^3 - e_1^2	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25

3) Аналогично производится анализ элемента 5, также сталкивающегося с необходимостью транспортирования события e_1^2 с инвертированным статусом по быстрой связи с элементом 11. На этот раз группа 2/11 разрешается с ложным статусом, поскольку все операнды завершились неудачно. От события e_1^2 отделяется расширение X_L^2, и событие, как неудачное, направляется по быстрой связи к элементу 25. Вторая группа не разрешается, поскольку требуется результат второго операнда. Событие e_1^2 уничтожается (табл. 6).

4)Производится анализ элемента 15, завершающийся быстрым транспортированием события e_1^3 с инвертированным статусом к элементу 24. Поскольку оператор AND признается удовлетворенным при первом же неудачным операнде, группа разрешается. При этом событие e_1^4 будет уничтожено вместе с расширением X_L^3 при первой же

попытке транспортирования, поскольку $\rho^{\mathrm{RESOLVED}}(X_L^3)=1$. Само событие e_1^3 направляется по быстрой связи к элементу 25, где также с негативным статусом разрешается группа X_L^1, и событие e_1^3 направляется к выходу модели.

Таблица 6. Состояние обработки ассерции на шаге №3

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^2	VAL = 0 RDEAD = 1 LDEAD = 1	25-in	∅	∅	OR2 NC=2 VAL=0 RESOLVED=1 JOINER: 25	∅
e_1^3	VAL = 1 RDEAD = 0 LDEAD = 0	15-in	e_1^3	e_1^4 - e_1^4	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 **NC=1** VAL=1 RESOLVED=0 JOINER: 25
e_1^4	VAL = 1 RDEAD = 0 LDEAD = 0	16-in	e_1^4	e_1^3 - e_1^3	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 **NC=1** VAL=1 RESOLVED=0 JOINER: 25

В течение единственного такта анализа описанный алгоритм потребовал только 7 операций обычного транспортирования и 3 операции быстрого транспортирования при разрешении групп. В результате применения метода иерархического разрешения групп статус вычисления первого вычислительного пути был получен с использованием только 15% возможных операций транспортирования (так как не потребовался параллельный анализ нескольких потоков активации), при этом 13 из 25 элементов модели не были задействованы при вычислениях. Можно утверждать о 6-кратном ускорении анализа для рассмотренного тестового воздействия (табл. 7).

Таблица 7. Состояние обработки ассерции на шаге №4

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^3	VAL = 0 RDEAD = 1 LDEAD = 0	25-in	∅	∅	AND2 NC=1 VAL=0 RESOLVED=1 JOINER: 24	OR2 NC=2 VAL=0 RESOLVED=1 JOINER: 25
e_1^4	VAL = 1 RDEAD = 1 LDEAD = 0	16-in	e_1^4	∅	AND2 NC=1 VAL=0 RESOLVED=1 JOINER: 24	OR2 NC=2 VAL=0 RESOLVED=1 JOINER: 25

Метод иерархического разрешения последовательностных групп, внося лишь минимальную дополнительную стоимость (новые связи в рамках событийных расширений, новый критерий очистки при итерировании контейнеров событий), обеспечивает максимально быстрое завершение анализа досрочно удовлетворяемых потоков активации. По сути, метод иерархического расширения групп развивает ранее предложенный метод предикторных связей на уровне пары соединитель-разделитель. При этом никак не изменяется стоимость обычного варианта анализа, при котором не наблюдается возможности досрочного удовлетворения вычислений.

4. Метод компрессии потока событий в бесконечной репетиции. Одним из контекстов, вызывающих значительное падение производительности анализа модели DRTLQ, является репетиция с бесконечной правой границей количества итераций. При длительном ожидании разрешения последующего потока репетиция генерирует и накапливает значительное число событий. Пусть имеется последовательность {a[*];b}, допуска-

ющая произвольное число повторений (от нуля до бесконечности) события $a=1$ перед возникновением события $b=1$. Структурное представление данной последовательности в модели DRTLQ показано на рис. 6. Во избежание создания и транспортирования на выход репетиции избыточных копий событий, которые при выполнении условия $b\neq 1$ будут уничтожены следующим элементом, здесь применяется ранее рассмотренная оптимизация обратного оповещения.

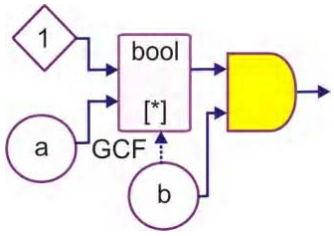

Рис. 6. Последовательность с репетицией с бесконечной правой границей

Предположим, на данную модель подается тестовое воздействие, представленное в табл. 8, в рамках которого в течение нескольких тактов инициируется входное событие $a=1$, а разрешающее событие $b=1$ наступает с некоторым опозданием. Исходя из семантики репетиций с бесконечными интервалами, репетиция накапливает события до момента разрешения.

Таблица 8. Работа функции-репетиции с бесконечной границей

#	a	b	Содержимое репетиции	Выход модели	Результат
1	1	0	$e_1^1\{r=1\}$	$\varnothing$	-
2	1	0	$e_1^1\{r=2\}$, $e_1^2\{r=1\}$	$\varnothing$	-
3	1	0	$e_1^1\{r=3\}$, $e_1^2\{r=2\}$, $e_1^3\{r=1\}$	$\varnothing$	-
4	1	0	$e_1^1\{r=4\}$, $e_1^2\{r=3\}$, $e_1^3\{r=2\}$, $e_1^4\{r=1\}$	$\varnothing$	-
5	1	1	$\varnothing$	e_1^1, e_1^2, e_1^3, e_1^4, e_1^5	$PASS^5$ $PASS^4$ $PASS^3$ $PASS^2$ $PASS^1$

Без применения оптимизации обратного оповещения количество событий, генерируемых элементом-репетицией, возрастало бы с квадратичной сложностью в зависимости от фактического количества тактов ожидания разрешения. Обратная связь от элемента-генератора b превращает данное количество в линейно возрастающую характеристику, однако при длительном ожидании может накопиться существенное количество событий.

Очевидным выводом из примера, рассмотренного в табл. 8, является тот факт, что при наступлении разрешающего события все накопленные данные транспортируются и обрабатываются далее совместно и одновременно, не покидая левой цепочки транспортирования. За исключением возможных дополнительных нюансов в ситуации нахождения события между парами соединитель-разделитель в логическом последовательностном операторе, все элементы цепочки событий либо одновременно достигнут выходов модели, либо будут одновременно уничтожены.

Данный факт предлагается использовать в целях уменьшения нагрузки на систему анализа ассерций. Сущность оптимизации состоит в применении метода компрессии цепочки событий в контексте разрешения функций-репетиций с бесконечными правыми интервалами. Всю цепочку накопленных событий можно заменить единственным событием с прикрепленным к нему специальным расширением, задающим множество моделируемых вычислительных потоков:

$$X_\infty :< X_{NEXT}, N_c, \{\rho, e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}^{\overline{i=0..(N_c-1)}} >, \quad (2)$$

где X_∞ – прикрепляемое событийное расширение компрессии; N_c – число сжатых событий; $\{\rho, e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}^{\overline{i=0..(N_c-1)}}$ – характеристический вектор и пара правых связей одного из событий, вошедших в сжатое расширение.

События могут быть сжаты в цепочку (2) только в том случае, если время создания каждого из них образует непрерывный ряд:

$$t_b(e_1) = t_b(e_2) - 1 = t_b(e_3) - 2 = \cdots = t_b(e_{N_c}) - N_c + 1. \quad (3)$$

Выполнение соотношения (3) обеспечивает возможность вычисления компонента t_b одного из сжатых событий, имея значение t_b для первого события (именно это значение ассоциируется с единственным событием, представляющим сжатую цепочку) и порядковый номер искомого события.

Характеристические векторы ρ^i могут различаться только в случае порождения сжатой цепочки между элементами соединитель-разделитель логического разветвления, когда альтернативный путь досрочно разрешает соответствующую группу. Поскольку данный факт представляется возможным установить статически во время подготовки модели, то в случае отсутствия окружающей логической группы цепочка (1) может не содержать элементов ρ ввиду их избыточности. Если же логическое разветвление имеется на родительском уровне модели, установление атрибута $\rho^{LDEAD} = 1$ на одном из событий, входящих в сжатую цепочку, не должно нарушать заложенный принцип компрессии. В тривиальных случаях можно также избежать присутствия в модели пары связей $\{e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}$, если поступающие на вход бесконечной репетиции события являются единственными в рамках представляемого ими потока активации.

При достижении ассерционного монитора или темпорального свойства событие, представляющее сжатую цепочку, требует специального режима обработки. Во-первых, если результат вычисления в соответствии разрешает поток активации, то речь идет об одновременном разрешении интервала потоков активации. Во-вторых, следует учитывать возможность наличия в сжатой цепочке некоторых записей о событиях с атрибутом $\rho^{LDEAD} = 1$, которые необходимо полностью проигнорировать.

Выводы. *Практическая значимость.* Предложенные модели и методы, составляющие основу программно-аппаратной среды верификации на основе ассерций, существенно (30%) повышают тестопригодность внутренних линий цифровой системы, что позволяет уменьшить временные затраты для создания теста, улучшить его функциональную полноту и качество проекта в целом.

Модели и методы функциональной верификации на основе темпоральных ассерций доведены до практической реализации в виде программных компонентов верификационной системы Riviera (Aldec Inc.), что дает возможность синтезировать специализированные и эффективные маршруты проверки и диагностирования цифровых систем на кристаллах.

Выигрыш производительности при применении предложенного метода компрессии событий на выходе бесконечных репетиций обусловлен следующими фактами:

1. Порождается меньшее количество событий. В худшем случае из каждого сжатого входного события сохраняется только тройка элементов $\{\rho, e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}$, в лучшем случае очередное входное событие, прикрепляемое к сжатой цепочке, лишь увеличивает значение счетчика N_c.

2. К выходам модели транспортируется только одно событие вместо цепочки из N_c событий. Данная характеристика производительности является константной относительно возрастания параметра N_c.

3. Специальные способы обработки требуются только на уровне ассерционного монитора и темпорального свойства. При этом возможность появления сжатой цепочки событий можно определить статически, что означает отсутствие каких-либо накладных расходов для формул, не содержащих элементы-репетиции с бесконечными интервалами.

Список литературы: 1. *Зайченко С.А., Хаханов В.И., Чумаченко С.В.* Структуры данных и модели реализации базовых элементов модели динамических регистровых очередей // Радиоэлектроника и информатика. 2010. № 1. С. 63-70. **2.** *Зайченко С.А., Чумаченко С.В.* DRTLQ-модель для функциональной верификации цифровых систем на основе линейной темпоральной логики // АСУ и приборы автоматики. 2010. Вып. 150. С. 33-48. **3.** *Berge J., O. Levia, J. Rouillard.* High-level System Modeling Specification Languages // Kluwer Academic Publishers, 1995, 162p. **4.** *Wakerly J.F.* Digital Design Principles and Practices. 1999. 895p. **5.** *Palnitkar S.* Verilog HDL: A Guide to Digital Design and Synthesis. Prentice Hall, 2003. 496p. **6.** *Lam W.K.*Hardware Design Verification: Simulation and Formal Method-Based Approaches. Prentice Hall, 2005, 585p. **7.** *Clarke E., Grumberg O., Peled D.* Model Checking. 6th edition, MIT Press, 2008. 313p. **8.** *Bening L.* An RTL Design Verification Linting Methodology // Proceedings of the International HDL Conference, 1999. P. 136-140. **9.** *Chakraborty R., Chowdhury D.R.* Raising the Level of Abstraction for the Timing Verification of System-on-Chips // IEEE Computer Society Annual Symposium on VLSI 2008, pp. 459-462. **10.** *Abramovici M., Breuer M.A., Friedman A.D.* Digital systems testing and testable design. Computer Science Press, 1998. 652 p. **11.** *Rabaey J.M., Pedram M.* Low-Power Design Methodologies // Kluwer Academic Publishers, 1996. 367p. **12.** *Piziali A.* Functional verification coverage measurement and analysis. Verisity Design Inc., Kluwer Academic Publishers, 2004. 213p. **13.** *Bening L., Foster H.* Principles of Verifiable Rtl Design: A Functional Coding Style Supporting Verification Processes in Verilog", Kluwer Academic Publishers, 2000. 281p. **14.** *Budkowski S., Najm E.* Formal Description Techniques and Protocol Specification, Testing and Verification. Chapman and Hall, 1998. 467p. **15.** *RTL* Design Style Guide for VHDL. English edition, STARC, 2003. 440p. **16.** *Melnyk D., Zaychenko S., Kolesnikov K., Lukashenko O.* Model of source code analyzer for hardware description languages // CAD Systems in Microelectronics, 2007. P. 113-114. **17.** *Foster H., Krolnik A., Lacey D.* Assertions-based Design. Kluwer Academic Publishers. 2003. 392p. **18.** *Assertion*-Based Verification. Synopsys Inc., Technical Report, May 2003. 14p. **19.** *Yeung P.* The Four Pillars of Assertions-Based Verification // Mentor Graphics Corporation, EuroDesignCon 2004. 21p.

Поступила в редколлегию 19.12.2009

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

Чумаченко Светлана Викторовна, д-р техн. наук, проф. кафедры АПВТ ХНУРЭ. Научные интересы: математическое моделирование, методы дискретной оптимизации. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326, e-mail: ri@kture.kharkov.ua.