УДК 681.324:519.613

С. А. ЗАЙЧЕНКО, И.Б. ЛЕШТАЕВА, В.Г. ВАРЧЕНКО, И.А. ЗЕНОВИЧ

МОДЕЛЬ ГЕТЕРОГЕННОГО ПОКРЫТИЯ ДЛЯ ЭФФЕКТИВНОЙ ФУНКЦИОНАЛЬНОЙ ВЕРИФИКАЦИИ ЦИФРОВЫХ СИСТЕМ

Рассматривается проблема разрозненности распространенных на практике метрик автоматического и функционального покрытия, применяющихся для измерения прогресса процесса функциональной верификации цифровых систем на кристаллах. Предлагается подход к управлению верификационным процессом на основе смешанных (гетерогенных) метрик покрытия, ориентированный на выявление наиболее существенных пробелов покрытия.

1. Введение

Согласно определению в работе [1], процесс функциональной верификации цифровых систем представляет собой демонстрацию сохранения в поведении фактической реализации системы требований спецификации и архитектурных намерений. Целью процесса функциональной верификации является приведение реализации системы в максимальное совпадение с ожидаемыми спецификацией внешне наблюдаемыми реакциями на воздействия, а также с интерпретацией требований в виде принятых внутренних решений и ограничений проектирования.

Достижение абсолютного качества системы в виде доказательства полного совпадения спецификации, архитектурных намерений и конкретной реализации для реальных промышленных проектов не представляется ни технически осуществимым, ни экономически целесообразным. Техническим барьером является нереалистичное для анализа количество всех возможных состояний системы, ограниченное вычислительными ресурсами и производительностью средств моделирования и верификации. В связи с этим, за исключением систем с критическими требованиями безопасности, необходимый уровень качества обуславливается в первую очередь экономическими ограничениями в виде "маркетинговых окон", скоростью эволюции требований пользователей к функциональности и наличия на рынке конкурирующих продуктов.

Очевидно, большее значение имеет достижение "приемлемого" для рынка уровня качества изделия [2]. Такой желаемый уровень качества задается специальным документом, называемым верификационным планом, представляющим собой конкретный конечный набор сценариев использования функциональности системы, способов подачи тестовых воздействий, механизмов наблюдения за реакцией системы, допустимых отклонений реализации. Выполнение пунктов верификационного плана отражает степень готовности проекта к более поздним низкоуровневым этапам проектирования. Определяемые сценарии должны быть заданы на достаточно высоком уровне абстракции, чтобы не создавать избыточных ограничений для группы проектирования, в то же время механизмы подачи воздействий и наблюдения реакций не обязаны взаимодействовать с реализацией как с "черным ящиком". Достаточный уровень абстракции верификационного плана способствует его повторному использованию по мере уточнения реализации системы.

Основными компонентами процесса функциональной верификации являются:

- формирование направленных либо ограниченных псевдослучайных тестовых воздействий на входы системы в соответствии с верификационным планом;
- считывание реакций системы через внешние выходы и дополнительные внутренние линии наблюдения (определенный верификационным планом расширенный интерфейс);
- сопоставление считанных реакций системы с заданными верификационным планом ожиданиями (scoreboard);
- проверка утверждений (ассерций) о корректности внешних интерфейсов компонентов и интегральности внутренних данных функциональных блоков;
- измерение прогресса верификации по метрикам покрытия (coverage metrics).

Основное внимание, как правило, уделяется достижению корректности работы функциональности, определенной верификационным планом. В то же время, не менее важным аспектом верификационного процесса является регулярное измерение метрик покрытия. Покрытие ортогонально по отношению к корректности функциональности и выявляет аспекты поведения системы, которым не было уделено должного внимания при составлении верификационного плана. Если такие методы верификации, как генерация воздействий, проверка ассерций, автоматизация тестовых сценариев, направлены на улучшение глубины диагностирования заданного набора сценариев, то метрики покрытия отражают достаточность верификационного процесса для достижения приемлемого уровня качества.

На практике применяется большое количество метрик покрытия, демонстрирующих прогресс верификационного процесса с тех или иных перспектив [3]. Принято разделять все метрики на две принципиально различные по происхождению категории: автоматически извлекаемые из описания модели устройства, или метрики покрытия кода (code coverage), и задаваемые пользователем вручную, называемые метриками покрытия функциональности (functional coverage). Метрики покрытия кода в большей степени отражают активизацию тестовыми сценариями блоков реализации, а метрики функциональности направлены на измерение активизации требований функциональности и архитектурных ограничений [4].

Ни одна из применяемых на практике метрик покрытия не дает гарантии достаточности верификационного процесса [5]. Более достоверный вывод о такой достаточности может быть сделан по совокупности результатов различных метрик. Комплексная разносторонняя оценка покрытия с применением одновременно нескольких метрик значительно уменьшает вероятность пропуска существенных пробелов в верификационном плане, за которыми могут скрываться критические дефекты. Для достижения качества необходимо построить верификационный процесс, прогресс которого измеряется как по автоматически извлекаемым из кода реализации системы показателям, так и по явно заданным критериям, вытекающим из спецификации и основных архитектурных решений.

Однако велика вероятность, что выявляемые пробелы покрытия, полученные во время анализа метрик различного типа, укажут на одинаковые ошибки, допущенные при интерпретации требований либо на более низком уровне при реализации системы [6]. Неясным остается принцип упорядочивания выявленных пробелов по различным метрикам покрытия. Ключевой вопрос формулируется следующим образом: какая именно из выявленных проблем покрытия наиболее существенно влияет на качество проекта и должна быть решена в первую очередь?

Еще одной проблемой анализа покрытия является разрозненность средств автоматизации верификационных маршрутов для различных метрик. Исторически программные комплексы верификации, автоматизирующие измерение конкретных метрик, развивались отдельно и не согласованы и несовместимы между собой. Задача унификации программных средств по анализу различных метрик покрытия в рамках единой согласованной методологии является предметом стандарта UCIS (Unified Coverage Interoperability Standard) [7].

Цель работы – существенное повышение эффективности функциональной верификации цифровых систем на ранних этапах проектирования за счёт выявления и упорядочивания пробелов в функциональном покрытии, наибольее влияющих на качество проекта.

Задачи исследования:

1) Анализ существующих активно применяющихся на практике метрик покрытия.
2) Создание математической модели унифицированного гетерогенного покрытия.
3) Отображение основных используемых метрик покрытия на предложенную унифицированную гетерогенную модель.
4) Создание структурной модели процесса функциональной верификации с применением унифицированного гетерогенного покрытия.

2. Унифицированная модель гетерогенного покрытия

Элементарной единицей анализа покрытия является точка покрытия P (coverage point) – некоторое состояние, переход, событие внутри верифицируемой системы, оказывающее, по мнению проектировщика, существенное влияние на ее функционирование. Для подтверждения полноты процесса функциональной верификации необходимо при помощи измерений во

время моделирования зафиксировать количество активизаций каждой из выбранных точек покрытия и сравнить их с заданными пороговыми значениями:

$$P = \{c, g, w\},\ C(P) = \begin{cases} 1, c \geq g, \\ 0, c < g, \end{cases} \quad (1)$$

где c – счетчик (counter) активизаций точки покрытия; g – заданное пороговое значение счетчика (goal); $C(P)$ – предикат, определяющий достаточную степень активизации точки покрытия P, а w – относительный безразмерный весовой коэффициент точки покрытия в общем покрытии.

При достижении счетчиком c порогового значения g дальнейший анализ точки покрытия можно отключить, поскольку последующие измерения не повлияют на общий результат.

Способ инкрементирования активизаций точки покрытия, выбор входных измеряемых атрибутов, необходимых для анализа точки, моменты времени для измерений, количество самих точек покрытия, выбор порогового значения полностью зависят от выбранной метрики.

Совокупность всех заданных точек покрытия формирует пространство покрытия:

$$S = (P_1, \ldots, P_N). \quad (2)$$

Анализ покрытия состоит в измерении доли $C(S)$ точек среди всех заданных точек покрытия, для которых во время моделирования зафиксировано достаточное количество активизаций:

$$C(S) = \frac{|P_i, C(P_i) = 1|}{|P_i|} \times 100\%, i = 1 \ldots N, \quad (3)$$

где числитель представляет собой количество всех точек покрытия, признанных активизированными, а знаменатель – общее число точек покрытия.

Достижение желаемой доли $C(S)$ наблюдаемых точек называют целью покрытия (coverage goal). Значение цели покрытия редко устанавливается на уровень 100%, поскольку достижение абсолютного покрытия выходит за рамки экономически приемлемых затрат.

Часто активизация одних точек покрытия является более важной, чем активизация других. Для обеспечения неравномерности влияния выбранных точек на общее покрытие пространства каждой точке ставится в соответствие относительный весовой коэффициент w_i. При этом задача измерения покрытия с учетом весов точек видоизменяется следующим образом:

$$C(S) = \frac{\sum_{i=1}^{N} C(P_i) \times w_i}{\sum_{i=1}^{N} w_i} \times 100\%. \quad (4)$$

Частным случаем является вес, равный 0, при этом имеет смысл отключить измерения соответствующей точки покрытия в целях экономии вычислительных затрат. При помощи нулевого веса часть точек покрытия можно исключить из анализа без повторного моделирования, что актуально для автоматически извлекаемых из HDL-описания метрик (например, для сгенерированного фрагмента). Исключением для формулы (4) является случай, при котором весы всех точек покрытия равны 0, тогда значением покрытия следует считать 100%.

Вес со значением 0 также можно использовать в ситуациях, когда один из видов покрытия на высоком уровне абстракции можно выразить через покрытие другого вида на более низком уровне абстракции. Безусловно, отдельная статистика от конкретной метрики покрытия может быть полезной для облегчения восприятия результатов анализа. Одна-

ко для повышения точности общих результатов и улучшения эффективности приоритезации проблем покрытия имеет смысл избегать при помощи нулевых весовых коэффициентов учета одних и тех же пробелов, представленных с нескольких точек зрения.

В целях удобства анализа пространство покрытия S может быть декомпозировано на некоторое множество субпространств $(S'_1,\ldots S'_M)$ (coverage scope), отражающих древовидную естественную иерархию блоков проекта. Формула (4) остается справедливой индивидуально для каждого субпространства. Общее рекурсивное покрытие пространства может быть вычислено из покрытий субпространств следующим образом:

$$C(S) = \frac{\sum_{k=1}^{M} C(S'_k) \times \sum_{i=1}^{N_k} w_{ki}}{\sum_{k=1}^{M} \sum_{i=1}^{N_k} w_{ki}} \times 100\%, \quad (5)$$

где $C(S'_k)$ – покрытие субпростнаства k; N_k – число точек покрытия в субпространстве k, а w_{ki} – весовой коэффициент точки покрытия i в субпространстве k.

Каждое из субпространств может также обладать индивидуальным весом W_k относительно других субпространств, что видоизменяет формулу (5) следующим образом:

$$C(S) = \frac{\sum_{k=1}^{M} (C(S'_k) \times \sum_{i=1}^{N_k} w_{ki} \times W_k)}{\sum_{k=1}^{M} (\sum_{i=1}^{N_k} w_{ki} \times W_k)} \times 100\%. \quad (6)$$

Глубина декомпозиции не ограничена, поскольку формулы (5) и (6) сохраняют справедливость независимо от наличия или отсутствия вложенных субпространств. Декомпозиция общего пространства покрытия имеет существенную значимость, поскольку стратегия ликвидации дефицита покрытия, т.е. движение от текущего уровня покрытия к цели покрытия, может быть различной в зависимости от типа анализируемой подсистемы.

Поскольку отдельные блоки в иерархии могут быть использованы более одного раза в различных режимах, при этом с идентичной реализацией, возникает потребность в вычислении агрегированного покрытия блока $C(\overline{S})$ за счет суммирования счетчиков из экземпляров:

$$C(\overline{S}) = \frac{\sum_{i=1}^{N} C(\overline{P}_i) \times w_i}{\sum_{i=1}^{N} w_i} \times 100\%, \quad (7)$$

$$C(\overline{P}) = \begin{cases} 1, (\sum_{j=1}^{N_{inst}} c) \geq t, \\ 0, (\sum_{j=1}^{N_{inst}} c) < t; \end{cases} \quad (8)$$

где $C(\overline{P})$ – предикат, определяющий достаточную степень активизации агрегированной точки покрытия по сумме счетчиков экземпляров блока, а N_{inst} – число экземпляров.

Вычисление агрегированного покрытия может приводить к существенно более низким вычислительным затратам по метрикам автоматического покрытия, в частности, для многократно используемых в проекте небольших идентичных блоков. Для небольшого

блока с четко заданными требованиями к функциональности более значимым является активизация всех его режимов в целом, нежели требование активизировать все возможные состояния, переходы, инструкции в каждом из экземпляров. Чем больше экземпляров блока в полной системе, тем выше вероятность ускорить достижение цели покрытия для данного блока, так как различные подсистемы вероятно будут использовать блок в различных режимах. Если такой дифференциации режимом использования не наблюдается, следует сделать вывод об избыточности функциональности блока, что также является важным выводом процесса функциональной верификации.

Анализ агрегированного покрытия имеет меньшую значимость для крупных подсистем, для которых многократное использование встречается реже. Агрегированное покрытие также не имеет существенного значения для функционального покрытия, поскольку применяется по отношению к крупномасштабным архитектурным единицам для которых характерно существенное индивидуальное влияние экземпляров на общее пространство покрытия.

Формула (6) может быть также применена для композиции в единый кумулятивный результат данных покрытия субпространств, относящихся к метрикам различного типа. В таком случае весовые коэффициенты W_k представляют собой средство приоритезации конкретных метрик на общий гетерогенный результат покрытия в целом. Кумулятивное гетерогенное покрытие является высокоуровневым обобщающим показателем, характеризующим полноту набора тестов и завершенность процесса функциональной верификации. При рациональной конфигурации весов конкретных метрик и пространств покрытия такой показатель четко демонстрирует достижимость проектной командой в рамках конкретных сроков заданного приемлемого уровня качества для выхода на рынок. Соответственно, кумулятивный результат покрытия может стоять в основе ряда важных управленческих решений о сроках, бюджете, кадровых и технических ресурсах проекта.

3. Автоматические метрики покрытия

Простейшим видом автоматического покрытия является покрытие инструкций (statement coverage). Каждой инструкции в HDL-коде ставится в соответствие точка покрытия, а соответствующий счетчик покрытия увеличивается, когда в выбранный момент времени поток управления моделирования проходит через данную инструкцию, т.е. при ее фактическом выполнении (табл. 1).

Таблица 1

Пример анализа покрытия инструкций

Statement 1	1	always @(posedge clk, negedge arst)
Statement 2	1	if (~arst)
		begin
Statement 3	0	Q <= 1'b0;
Statement 4	0	QN <= 1'b1;
		end
Statement 5	1	else begin
Statement 6	1	Q <= data;
Statement 7	1	QN <= ~data;
		end

Вариацией данной метрики является покрытие строк HDL-описания (line coverage), являющееся менее точной метрикой, поскольку каждая линия может теоретически содержать несколько инструкций.

Более компактной аналогичной метрикой является покрытие блоков инструкций (block coverage), при которой одна точка покрытия ставится в соответствие последовательности инструкций, не содержащей условных (if, case), циклических (for, while, repeat) с переменным количеством итераций и временных разветвлений (wait), а также вызовов подпрограмм. Отсутствие разветвлений в последовательности инструкций означает эквивалентность статистики их активизации (за исключением ручного прерывания процесса моделирования, чем можно пренебречь). За счет уменьшения количества измеряемых точек покрытие блоков измеряется с существенно меньшими затратами производительности (табл. 2). Обладая структурной информацией об инструкциях, начинающих и завершающих

блоки, покрытие инструкций может быть приведено к покрытию блоков элементарным преобразованием без повторного моделирования с помощью следующих отношений:

$$C(P_{block}) = C(P_{stmt-1}),\ w_{block} = \frac{\sum_{i=1}^{N} w_{stmt-i}}{N}, \quad (9)$$

где N – число инструкций внутри блока, а P_{stmt-1} – точка покрытия любой из инструкций, входящих в блок. Покрытие блока есть покрытие субпространства из составляющих блок инструкций, при этом вес первой инструкции соответствует весу блока, а вес остальных равен 0.

Таблица 2

Пример анализа покрытия блоков

Block-1	1	always @(posedge clk, negedge arst)
		if (~arst)
Block-2	0	begin
		Q <= 1'b0;
		QN <= 1'b1;
		end
Block-3	1	else begin
		Q <= data;
		QN <= ~data;
		end

Правила преобразования одних видов автоматического покрытия в другие обеспечивают возможности для оптимизации вычислительного цикла измерений. Очевидно, что при прочих равных условиях, предпочтительней является метрика покрытия, использующая меньшее количество счетчиков и измерений.

Важной автоматической метрикой является покрытие разветвлений (branch coverage), при которой в качестве точки покрытия выступает путь выполнения при проходе через условную конструкцию. Следует отметить, что даже при отсутствии указанных в явном виде путей выполнения для несоблюдения всех условий (путь else для инструкции if, путь default для инструкции case) точка покрытия для этих путей должна быть создана (табл.3).

Таблица 3

Пример анализа покрытия разветвлений

Block-1			always @(data or en)
	Branch scope 1	0t/1f	if (en)
Block-2	Branch path 1-1	0	latch_output <= data;
	Branch path 1-2	1	// no explicit else

Покрытие разветвлений можно выразить через эквивалентное покрытие блоков. Для явно заданных в HDL-коде путей разветвления достаточно использовать счетчик покрытия блока дочерней инструкции. Для неявных альтернативных путей, называемых “all false”, следует вычесть из счетчика родительского блока сумму значений счетчиков всех явно заданных путей:

$$C(P_{BP1-1}) = C(P_{block-2}),\ C(P_{BP1-2}) = \begin{cases} 1, (c_{block-1} - c_{block-2}) \geq g_{BP1-2}, \\ 0, (c_{block-1} - c_{block-2}) < g_{BP1-2}. \end{cases} \quad (10)$$

Соответственно, если покрытие блоков/инструкций и покрытие разветвлений анализируются одновременно в рамках одного сеанса моделирования, в целях экономии вычислительных ресурсов не следует проводить двойное компилятивное инструментирование генерируемого для HDL-симулятора машинного кода в целях анализа каждой из рассматриваемых метрик. Вместо этого следует ограничиться подготовкой модели для анализа покрытия блоков/инструкций, а покрытие разветвлений подсчитать используя (10) из собранных данных блоков/инструкций.

Более сложный случай составляют вложенные условия, где подобные вычитания нужно применять последовательно к каждому из уровней вложенности условий (табл.4).

$$C(P_{BP1-1}) = C(P_{block-2}),\ C(P_{BP2-1}) = C(P_{block-4}),$$

$$C(P_{BP1-2}) = C(P_{BLOCK-3}) = \begin{cases} 1, (c_{block-1} - c_{block-2}) \geq g_{BP1-2}, \\ 0, (c_{block-1} - c_{block-2}) < g_{BP1-2}, \end{cases}$$

$$C(P_{BP2-2}) = \begin{cases} 1, (c_{block-3} - c_{block-4}) \geq g_{BP2-2}, \\ 0, (c_{block-3} - c_{block-4}) < g_{BP2-2}, \end{cases} \quad (11)$$

$$C(P_{BP1-3}) = \begin{cases} 1, (c_{block-1} - c_{block-2} - c_{block-3}) \geq g_{BP1-3}, \\ 0, (c_{block-1} - c_{block-2} - c_{block-3}) < g_{BP2-2}. \end{cases}$$

Таблица 4

Пример анализа покрытия вложенных разветвлений

Block-1		3	process (clk, arst) begin
	Branch scope 1		if(arst = '0') then
Block-2	Branch Path 1-1	1	res <= '0';
	Branch Path 1-2		elsif (clk'event and clk='1') then
Block-3	Branch scope 2	2	if (en = '1') the
Block-4	Branch Path 2-1	1	res <= data;
	Branch Path 2-2		end if;
	Branch Path 1-3		end if; end process;

(11)

Еще одной базисной метрикой является покрытие условных выражений (condition/ expression coverage), которое не представляется возможным выразить как проекцию покрытия инструкций. Условные переходы в RTL-кодировании могут контролироваться сложными управляющими выражениями, объединяющими несколько арифметико-логических термов на векторных и скалярных входных данных, при помощи логических операторов. Недостаточно убедиться в активизации инструкций, выполняющихся при попадании в разветвленную ветвь. Необходимо также удостовериться, что активизированы все возможные комбинации управляющих термов, которые разрешают либо запрещают выполнение ветвей. За отсутствием активизации некоторых комбинаций могут скрываться функциональные ошибки, не позволяющие привести тот или иной управляющий терм в необходимое состояние. Также пробелы в покрытии управляющих выражений могут указывать на его избыточность, в том числе на использование термов, которые не могут быть активизированы ни при каких условиях.

Точки покрытия управляющих выражений формируются в соответствии с таблицей истинности, где элементарными входными переменными считаются арифметико-логические термы. Создавать отдельную точку покрытия для каждой строки таблицы истинности не имеет смысла, поэтому таблицу следует привести к минимальной кубической форме любым способом. В примере из табл. 5 для достижения 100% покрытия управляющего выражения достаточно активизировать 4 существенные комбинации термов.

Таблица 5

Пример анализа покрытия условных выражений

	a	b	c	a&b|c
if (a & b | c)	0	-	0	0
…	-	0	0	0
	-	-	1	1
	1	1	-	1

Существует также альтернативная метрика анализа покрытий условных выражений, называемая метрикой чувствительности условий (sensitized/focused condition coverage).

Ключевое внимание здесь заостряется на факторах изменения результата условия в целом. Отличие от изложенного выше табличного способа состоит в создании точек покрытия для тех комбинаций значений термов, в которых изменение значения одного из термов повлияет на значение функции в целом. Это можно свести к взятию частных производных от управляющей функции по каждой из входных переменных – формула (12) и выписыванию тех комбинаций остальных переменных, при которых частная производная равна 1 (табл. 6):

$$\frac{d(a\,\&\,b\mid c)}{d(a)} = (0\,\&\,b\mid c)\oplus(1\,\&\,b\mid c) = c\oplus(b\mid c) = b\wedge\overline{c},$$

$$\frac{d(a\,\&\,b\mid c)}{d(b)} = (a\,\&\,0\mid c)\oplus(a\,\&\,1\mid c) = c\oplus(a\mid c) = a\wedge\overline{c}, \qquad (12)$$

$$\frac{d(a\,\&\,b\mid c)}{d(c)} = (a\,\&\,b\mid 0)\oplus(a\,\&\,b\mid 1) = (a\,\&\,b)\oplus 1 = \overline{a}\vee\overline{b}.$$

Таблица 6

Анализ покрытия чувствительности условий

`if ( a & b | c )` `…`	По a: $b\wedge\overline{c}$: Точка покрытия №1: {0,1,0} Точка покрытия №2: {1,1,0} По b: $a\cdot\overline{c}$ Точка покрытия №3: {1,0,0} Точка покрытия №4: {1,1,0} По c: $\overline{a}\vee\overline{b}$ Точка покрытия №5: {0,-,0} {-,0,0} Точка покрытия №6: {0,-,1} {-,0,1}

Управляющая логика большинства реальных цифровых изделий, как правило, содержит один или несколько конечных автоматов, и как источник многих видов функциональных нарушений является объектом пристального внимания при верификации. В верификационный план типично попадают следующие требования к покрытию конечных автоматов:

– автомат должен хотя бы однажды побывать в каждом из состояний (state coverage);

– каждый возможный переход между состояниями автомата должен быть активизирован хотя бы однажды (transition coverage);

– каждое возможное условие перехода из имеющихся альтернатив (дуги переходов) должно быть активизировано хотя бы однажды (arc coverage).

Число стилей описания конечных автоматов, применяемых на практике при RTL-кодировании, весьма ограничено, поскольку проектировщики стремятся использовать максимально стандартные типы описаний, распознаваемые большинством инструментов логического синтеза. Типичный стиль описания автомата включает конструкцию case, в которой каждая из ветвей описывает логику перехода и присвоения выходов (иногда в отдельных процессах) для каждого из состояний автомата. Для компиляторов не представляет сложности распознать в таком описании регистры состояний, инструкции переходов, а также управляющие условия переходов.

При соответствующей предварительной подготовке структурной информациис помощью статистики от базисных метрик возможно измерить покрытие моделей конечных автоматов без создания специальных проверяющих моделей. Для подсчета покрытия состояний и переходов достаточно применения покрытия инструкций, и только для более детализированного анализа покрытия дуг необходимы дополнительные данные от анализа покрытия управляющих выражений. Спроецировав на распознанную в RTL-коде структурную модель конечного автомата соответствующие данные покрытия от базисных метрик, без дополнительных затрат можно достичь высокоуровневых целей верификационного плана (табл.7).

Таблица 7

Анализ покрытия состояний и переходов конечного автомата

`enum { stR, st0, st1 } state;`
 	
`always @(posedge clk or posedge reset)`
`begin`
`  if (reset)`
`    state = stR;`
`  else`
`    case(state)`
`      stR:`
`        if (i==0)`
`          state = st0;`
`        else`
`          state = st1;`
`      st0:`
`        if (i%2)`
`          state = st1;`
`      st1:`
`        state = st0;`
`    endcase`
`end`
	Block-1
	
	Block-2
	
	Block-3
	
	Block-4
	
	Block-5
		$c_{STATE-stR} = c(P_{block-1})$
		$c_{STATE-st0} = c(P_{block-2}) + c(P_{block-5})$
		$c_{STATE-st1} = c(P_{block-3}) + c(P_{block-4})$
		
		$c_{TR-stR} = c(P_{block-1})$
		$c_{TR-stR-st0} = c(P_{block-2})$
		$c_{TR-stR-st1} = c(P_{block-3})$
		$c_{TR-st0-st1} = c(P_{block-4})$
		$c_{TR-st1-st0} = c(P_{block-5})$

Несколько обособленно стоит метрика покрытия переключений однобитных сигналов (toggle coverage), демонстрирующая, наблюдались ли на каждом из сигналов во время моделирования переходы между значениями: 0->1, 1->0. В ряде специализированных случаев, например, для 3-значной логики с применением Z-состояния, интересуют также дополнительные переходы 0->Z, 1->Z, Z->0, Z->1. Применение данной метрики покрытия выявляет наличие в высокоуровневой модели ошибок, внешне подобных анализу константных неисправностей в физических чипах (stuck-at-fault). Разумеется, на RTL-уровне причинами таких ошибок являются не дефекты изготовления, а неправильное соединение портов блоков или отсутствие драйвера сигнала.

Покрытие переключений является дорогостоящей для анализа метрикой, в частности, следует с осторожностью применять ее для больших векторных сигналов, особенно для блоков памяти, поскольку это может привести к существенной деградации производительности моделирования. Высокая стоимость такого вида анализа обуславливается необходимостью внедрения дополнительных искусственных процессов, чувствительных к изменениям каждого из сигналов. В результате необдуманного применения данной метрики к большому количеству сигналов число процессов в модели может существенно возрасти, что выразится в уменьшении производительности моделирования на 2-4 порядка.

Учитывая возможности гетерогенной модели покрытия, когда большинство из ключевых причин недостаточности переключений между сигналами может быть выявлено с применением менее дорогостоящих метрик (например, невыполненный код), рекомендуется ограничивать применение данного типа покрытия следующими случаями:

– входные порты модуля (с приоритетом для управляющих вводов);
– выходы tri-state буферов;
– двусторонние порты;
– выходы дочерних блоков, особенно для экземпляров внешних IP.

Большинство проблем, выражающихся в отсутствии переключений сигналов, являются структурными ошибками соединений. Если помимо симуляции в верификационном цикле используется статический структурный анализ (линтинг), необходимость в анализе покрытия переключений практически отпадает. Выявление некорректности соединения при помощи чисто статического анализа без применения моделирования имеет на несколько порядков меньше вычислительных затрат по сравнению с анализом покрытия переключений.

4. Функциональные метрики покрытия

Анализ метрик функционального покрытия более сложный по сравнению с анализом метрик автоматического покрытия. Проверяющая модель не извлекается автоматически, а требует ручного определения на основе требований спецификации и значительных интеллектуальных усилий.

Методология функциональной верификации на основе темпоральных ассерций предусматривает использование статистики активизации проверяемых логико-временных утверждений в качестве иллюстративной метрики функционального покрытия. Для каждого начального момента моделирования ассерция может находиться в одном из перечисленных состояний {0, 1, X}:

1. Ассерция активизирована и удовлетворена (pass = 0).
2. Ассерция активизирована и нарушена (fail = 1).
3. Ассерция не активизирована (not activated = X).

Ассерции фактически представляют собой форму записи требований спецификации с применением формального синтаксиса. Результаты pass/fail важны для оценки функциональной корректности работы проектируемой системы. Независимо от показателя корректности, статус pass/fail можно считать положительным с точки зрения анализа покрытия, поскольку, очевидно, верифицируемое поведение затрагивается имеющимся набором тестов. Статус not activated нельзя считать положительным для анализа покрытия, потому что по той или иной причине функциональность не проявляется при тестировании. Проблемной с точки зрения анализа покрытия является ситуация, когда ассерция имеет статус not activated на всех вычислительных путях во всех тестах. 100% уровень ассерционного покрытия достигается тогда, когда в системе не остается ни одной ассерции, не активированной хотя бы одним тестом:

$$\forall k = 1..q, \bigcap_{i=1}^{p} (A_{ki}(t) \neq X) = 0, \tag{13}$$

где q – число всех ассерций; p – число тестов, а $A_{ki}(t)$ – обобщенная реакция ассерции с порядковым номером $k \in [1; q]$ на тест с порядковым номером $i \in [1; p]$ на всех путях.

Таким образом, количество срабатываний индивидуальной заданной ассерции $c(A_k)$ можно определить как количество вычислительных путей π_t от начального t_{beg} до конечного t_{end} момента моделирования, на которых статус анализа ассерции отличается от not activated:

$$\forall t = t_{beg}..t_{end}, c(A_k) = \left| \pi_t, A_k(t) \in \{0,1\} \right|. \tag{14}$$

Если активизация ассерции не фиксируется ни разу за все время моделирования на всех имеющихся тестах, следует провести диагностику и локализовать источник проблемы. Описание ассерции может содержать ошибку в части предусловия последовательностной импликации, что блокирует переход к анализу основной части верифицируемого темпорального утверждения (vacuous pass). Если ошибки не обнаружено, отсутствие активизации говорит о существенном пробеле покрытия в тестовом наборе, поскольку тесты не воздействуют на одно из автоматизированных в виде ассерционного описания требований спецификации. Также отсутствие факта активизации ассерции при моделировании может означать избыточность/неактуальность определенной ассерции.

В языках PSL и SystemVerilog существует специальная форма описания ассерций – cover-директивы, ориентированная на анализ покрытия, а не на проверку функциональной корректности. Такие директивы могут содержать темпоральное утверждение, включающее логический и последовательностный уровни. В cover-директивах нельзя использовать высокоуровневые темпоральные операторы уровня свойств, такие как последовательностная импликация или операторы с семантикой глобального времени. В связи с этим cover-директива не может иметь статуса not activated.

Выявление в системе успешного выполнения интересующей последовательности событий, принадлежащего некоторой cover-директиве $A^{COVER}{}_k$, может быть использовано в

качестве точки покрытия. В свою очередь, в отличие от традиционного типа ассерций неудачное выполнение последовательности в cover-директиве не оказывает никакого влияния на результаты анализа:

$$\forall t = t_{beg}..t_{end}, c(A^{COVER}{}_k) = \left|\pi_t, A_k(t) = 1\right|. \quad (15)$$

Наиболее специализированный характер носит анализ конструкций covergroup из языка SystemVerilog. Данный блок языковых конструкций предназначен исключительно для анализа функционального покрытия и не имеет другого назначения. Конструкции covergroup удачно вписываются в предложенную модель унифицированного гетерогенного покрытия, при этом понятию точка покрытия соответствуют простые и перекрестные "корзины" (bins), а конструкции coverpoint и cross моделируются как вложенные в пространство покрытия covergroup субпространства. Использование конструкций covergroup имеет ряд уникальных по сравнению с остальными видами покрытия возможностей:

– полный процедурный контроль над процессом анализа покрытия со стороны тестового окружения при помощи стандартных системных функций и функций-методов, в том числе полная или частичная остановка и возобновление, получение текущей мгновенной статистики покрытия без прерывания моделирования, форсирование считывания данных;

– явная возможность определения весовых коэффициентов пространств и субпространств при помощи HDL-кода, в том числе изменение в ходе моделирования;

– гибкие возможности контроля над моментами считывания данных;

– возможность определения обобщенной схемы функционального покрытия за счет передачи фактических интересующих сигналов через формальные аргументы-ссылки;

– возможность определения неравномерных интервалов значений для точек покрытия, в том числе при помощи кубической формы (wildcard);

– богатые возможности фильтрации нежелательных данных.

Конструкции covergroup имеют объектно-ориентированный стиль, что, в отличие от механизма ассерций и других метрик, позволяет анализировать покрытие для динамических объектов, контролировать состав списка интересующих значений и событий во время моделирования. Такой стиль способствует созданию интеллектуальных тестовых сред и повторно используемых верификационных библиотек, таких как OVM/UVM.

В результате подготовки структур данных при создании объекта-covergroup каждой сформированной корзине ставится в соответствие специальная логическая функция, определяющая правило инкрементирования счетчика покрытия:

$$c(P_{bin-i}) = \sum_{k=1}^{N_{SAMPLE}} F_i(x(t_k)) \wedge \overline{R_i(t)}, \quad (16)$$

где i – номер корзины; N_{SAMPLE} – число итераций считывания интересующего входного выражения x в моменты времени t_k, соответствующие считываниям, инициированным вручную или автоматически; F_i – функция, разрешающая инкрементирование в зависимости от особенностей определения значений, входящих в корзину, а R_i – необязательная функция-фильтр, возвращающая значение 1, когда считывание должно быть проигнорировано системой анализа.

В простейшем случае, когда не определено специальных правил разделения диапазона возможных значений выражения по корзинам, симулятор автоматически распределяет между корзинами весь диапазон, разделяя его на равные интервалы значений. При автоматическом разбиении число корзин определяется конфигурацией симулятора и обычно не превышает 64 для всех типов данных. Формируемая логическая функция инкрементирования при этом сводится к простейшему сравнению фактического считанного значения с границами заданного интервала.

Поскольку наборы значений интересующего выражения в общем случае могут пересекаться между несколькими корзинами, один шаг анализа предполагает считывание текущего значения выражения, а затем проверку функции инкрементирования для каждой из корзин.

Немаловажную роль также играют специальные перекрестные корзины (cross bins), представляющие собой декартово произведение множеств корзин от нескольких (двух и более) входящих в объект-covergroup входных выражений. Специальные множественные операторы семейства binsof позволяют определить подмножества кортежей, образуемых декартовым произведением корзин, относящиеся к перекрестным корзинам:

$$c(P_{cross-bin-i}) = \sum_{k=1}^{N_{SAMPLE}} G_i(x_1(t_k),\ldots,x_M(t_k)) \wedge \overline{R_i(t)}, \qquad (17)$$

где G_i представляет собой M-арный предикат, который соответствует перекрестной корзине с номером i, определяющий принадлежность к ней кортежа.

При отсутствии пользовательских правил распределения кортежей по корзинам формируется автоматическая корзина для каждого возможного кортежа. Поскольку количество перекрестных корзин в этом случае может быть очень большим, симуляторы избегают создания объектов-корзин, пока значение соответствующих счетчиков не станет положительным. Это предполагает использование структур данных для хранения корзин, основанных на разреженных матрицах.

5. Структурная модель процесса верификации на основе гетерогенного покрытия

Целью процесса функциональной верификации, основанного на анализе покрытия при помощи предложенной унифицированной гетерогенной модели, является достижение запланированного приемлемого порогового значения кумулятивного покрытия в соответствии с (6) с наименьшими временными затратами. Унификация модели для различных метрик покрытия позволяет создать более эффективный процесс путем устранения ряда проблем:

1. Без унификации модели результаты анализа покрытия сохраняются в базах данных различных несовместимых форматов, что обуславливает разрозненность инструментов анализа, генерируемых отчетов, процедурного доступа. Унифицированная модель открывает возможность для создания единого формата результирующей базы покрытия, что способствует созданию согласованного маршрута верификации, консистентных инструментов, отчетов, единого процедурного интерфейса для прикладных программ.
2. Разрозненные метрики кодового покрытия предполагают наличие несвязанных механизмов инструментирования, необходимого для организации автоматического сбора данных о покрытии во время моделирования. На основании выведенных соотношений между различными метриками покрытия уницифированная модель позволяет уменьшить объемы инструментирования при одновременном анализе метрик, что снижает вычислительные затраты на подготовку моделии и на время непосредственного моделирования.
3. Применяя метрики покрытия по отдельности, можно выявить одинаковые пробелы в покрытии с различных перспектив. В результате процесс устранения пробелов затягивается, а регрессионный тестовый набор становится избыточным. Унифицированная модель в сочетании с ее предварительным структурным анализом позволяет упростить процесс ликвидации пробелов покрытия с помощью гибкой расстановки весовых коэффициентов, надежно фильтрующих из рассмотрения проблемы-дупликаты.

Предлагаемый процесс функциональной верификации представлен на рисунке.

Сначала исходное HDL-описание обрабатывается HDL-компилятором, при этом оптимизации HDL-модели отключаются во избежание искажений результатов анализа покрытия. В процессе компиляции внедряется система инструментирования, которая в зависимости от конфигурации и включенных метрик генерирует минимально возможную избыточность двоичного кода, необходимую для сбора данных покрытия во время моделирования.

Одной из важнейших задач системы инструментирования является конфигурируемая автоматическая расстановка нулевых весовых коэффициентов по результатам структурного анализа. Предпочтение отдается высокоуровневым метрикам функциональной верификации. В частности, к типичным автоматизируемым процедурам данной направленности относятся:

– включение агрегатного режима для многократно использованных в иерархии модулей проекта и выключение иерархического режима для однократно использованных модулей;

– отключение влияния инструкций и разветвлений, формирующих описания состояний и переходов конечных автоматов;

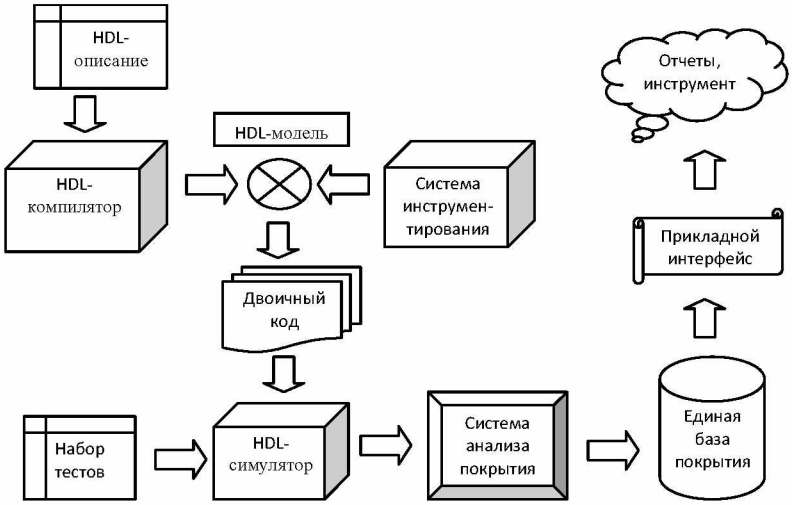
– отключение покрытия переключений для сигналов, наблюдаемых конструкцией covergroup;

– минимизация количества счетчиков покрытия инструкций для процессов с эквивалентными списками чувствительности;

– минимизация количества счетчиков покрытия разветвлений для процессов с эквивалентными списками чувствительности и структурой условных выражений (эффективно для типовых схем синхронизации, сброса регистров);

– автоматический выбор между анализом термов управляющих выражений и анализом разветвлений в зависимости от оценки сложности термов;

– отключение всех автоматических метрик покрытия для блоков, для которых характерен высокий уровень плотности ассерционных описаний.



Структурная модель процесса верификации на основе гетерогенного покрытия

Результирующий инструментированный двоичный код модели вместе с набором тестовых воздействий поступает в HDL-симулятор. По ходу моделирования симулятор, выполняя подготовленный двоичный код, взаимодействует с системой анализа покрытия, регистрируя собранные данные от различных метрик.

Результаты анализа покрытия записываются в единую унифицированную базу данных. Дальнейшая обработка данных может осуществляться вне сеанса моделирования. Для получения данных о покрытии предоставляется специальный унифицированный прикладной интерфейс на языке программирования C, совместимый с предварительным стандартом Accellera UCIS. Прикладной интерфейс может быть использован для интеграции графических инструментов отладки, генерации различных отчетов, статистического анализа, сравнения изменений результатов в цикле регрессионного тестирования и других актуальных аналитических задач.

Набор тестов для современных цифровых систем может оказаться внушительным, и для их регулярного запуска с сохранением быстроты отклика верификационного процесса часто применяют параллельный запуск тестов на вычислительном кластере. В качестве кластера может выступать как внутренняя GRID-система проектной организации, так и ряд специализированных публичных вычислительных “облак”. Каждый отдельный тест в таком случае формирует отдельную результирующую базу покрытия. Необходимость в проведении дальнейшей обработки данных, генерации отчетов предполагает возможность слияния нескольких результатов моделирования в единую базу. Такая операция имеет важное практическое значение и должна предоставляться прикладным интерфейсом базы покрытия.

6. Выводы

В ходе исследования были решены поставленные *задачи*:

1) Проанализированы существующие активно применяющиеся на практике метрики автоматического и функционального покрытия.

2) Предложена математическая модель унифицированного гетерогенного покрытия, обобщающая все имеющиеся виды метрик в единую структуру при помощи универсальных понятий точки и пространства покрытия.

3) Показано отображение используемых метрик покрытия на предложенную унифицированную модель.

4) Разработана структурная модель процесса функциональной верификации с применением унифицированного гетерогенного покрытия.

Главный *научный результат* исследования состоит в формулировке унифицированной модели гетерогенного покрытия, позволяющей синтезировать согласованные верификационные маршруты, основанные на комплексных автоматических и функциональных метриках.

Ключевым *практическим результатом* исследования является существенное повышение эффективности функциональной верификации на ранних этапах проектирования за счёт выявления и упорядочивания пробелов в функциональном покрытии, наиболее влияющих на качество проекта, путем снижения вычислительных затрат, связанных с зависимыми метриками покрытия.

Унификация обработки различных видов покрытия в единой гетерогенной модели открывает путь к созданию программных средств автоматизации, обладающих простотой, согласованностью, однозначностью выводов, что способствует качественным улучшениям в процессе функциональной верификации.

Список литературы: 1. *Piziali A.* Functional Verification Coverage Measurement And Analysis // Norwell: Springer, 2004. 213 p. **2.** *Feierbach G., Gupta V.* True Coverage: A Goal of Verification // Proc. of the Fourth International Symposium on Quality Electronic Design (ISQED'03). 2003. P. 75 – 78. **3.** *Jerini'c V., Langer J., Heinkel U., Müller D.* New Methods and Coverage Metrics for Functional Verification // IEEE Conference Publications. 2006. P.1-6. **4.** *Spear C.* SystemVerilog for Verification: A Guide to Learning the Testbench Language Features // Springer, 2008. 429 p. **5.** *Verma S., Harris I., Ramineni K.* Automatic Generation of Functional Coverage Models from Behavioral Verilog Descriptions // Design Automation Test in Europe Conference Exhibition. 2007. P. 3-8. **6.** *Asaf S., Marcus E., Ziv A.* Defining Coverage Views to Improve Functional Coverage Analysis", Proceedings of the 41st Design Automation Conference (DAC'04). 2004. P.41-44. **7.** *Ho R., Sarkar A.* An Introduction to the Unified Coverage Interoperability Standard // DVCon 2012. 111p.