Функциональная верификация цифровой аппаратуры

Моделирование: сбор тестового покрытия в Verilator

Фролов П.В.

November 26, 2025

Компиляция тестового стенда для сбора тестового покрытия

Информация на слайдах проверялась для `Verilator 5.042`.

Для сбора тестового покрытия необходимо собрать тестовый стенд с одним ключей

- `--coverage-line` инструкции (строки кода);
- `--coverage-expr` выражения;
- `--coverage-toggle` переключения;
- `--coverage-user` описание функционального покрытия (`cover`) ;
- `--coverage`

Запуск моделирования со сбором тестового покрытия

- При запуске моделирования тестового стенда тестовое покрытие собирается в файле `coverage.dat`.
- Чтобы изменить имя выходного файла с собранным покрытием, необходимо запустить моделирование тестового стенда с опцией:
  `+verilator+coverage+file+<выходной_файл_с_покрытием>`

Обработка файлов с собранным тестовым покрытием

- Слияние данных о покрытии, полученных в результате нескольких запусков на одном и том же RTL-описании, в один выходной файл.

```
verilator_coverage --write <выходной_файл_покрытия> <входной_файл_покрытия_0> \
        [<входной_файл_покрытия_1> \
        [ <входной_файл_покрытия_2> ..]]
```

- Формирование отчёта о покрытии в виде аннотированных файлов с исходным кодом тестового стенда.

```
verilator_coverage --annotate-all \
        --filter-type <метрика_покрытия> \
        --annotate-points \
        --annotate-min 1 \
        --annotate <директория_с_отчётами> <входной_файл_покрытия>
```

```
метрика_покрытия := toggle|line|branch|expr|user|<wildcard with *, ?>;
```

Расшифровка аннотации

- Каждая строка исходного кода может описывать несколько точек (элементов) тестового покрытия.
- При моделировании подсчитывается число исполнений каждого элемента тестового покрытия: M.
- При аннотировании данные о тестовом покрытии вставляются в начало соответствующих им строк исходного кода в формате
  `<специальный символ><M: число задействования элемента покрытия при моделировании>`.
- Для аннотации строки, содержащей несколько элементов тестового покрытия, используются минимальное M_{max} и максимальное M_{min} значения среди этих элементов. При этом для строки в аннотацию попадает $M = M_{max}$.
- Специальный символ означают отношение числа справа от него к пороговому значению параметра `--annotate-min` команды `verilator_coverage` и могут быть использованы для построчной фильтрации отчёта о покрытии (например, с помощью команды `grep`).
  - (пробел): для всех точек покрытия в строке порог достигнут;
  - % всех точки покрытия в строке не достигли порога;
  - ~ некоторые точки покрытия в строке достигли порога, некоторые – нет;
  - + точка покрытия достигла порога (используется только при ключе `--annotate-points`);
  - - точка покрытия не достигла порога (используется только при ключе `--annotate-points`).
- Для аннотирования каждого элемента покрытия необходимо подать ключ `--annotate-points`.

Генерация отчёта о тестовом покрытии

Рекомендации

- Рекомендуется формировать отчёты о покрытии в разных метриках по отдельности.
- Метрику строк (исполняемых выражений) verilator разбивает на две метрики: строк `line` и ветвлений `branch`. Отчёт о покрытии в метрике исполняемых выражений следует формировать с помощью ключа `--filter-type="*n*"` (`branch` и `line`).
- Для метрики строк/ветвлений ключ `--annotate-points` нужен при наличии в тестовом стенде нескольких экземпляров тестируемых модулей.

Отчёт о тестовом покрытии

Строки кода и ветвления (lines, branches)

```
 000001        normalise_b:
 000001        begin
~000001          if (b_m[52]) begin
 000001            state <= divide_0;
%000000          end else begin
%000000            b_m <= b_m << 1;
%000000            b_e <= b_e - 1;
                 end
               end
```

Отчёт о тестовом покрытии

Переключения (toggle)

```
 000114   reg       [3:0] state;
+000114  point: comment=state[0]:0->1 hier=top_module.dd
+000114  point: comment=state[0]:1->0 hier=top_module.dd
+000111  point: comment=state[1]:0->1 hier=top_module.dd
+000110  point: comment=state[1]:1->0 hier=top_module.dd
+000109  point: comment=state[2]:0->1 hier=top_module.dd
+000108  point: comment=state[2]:1->0 hier=top_module.dd
+000108  point: comment=state[3]:0->1 hier=top_module.dd
+000107  point: comment=state[3]:1->0 hier=top_module.dd
```

Отчёт о тестовом покрытии

Выражения (expressions)

```
%000004          if ((~t1 && t2) || (~t3 && t4)) $write("");
-000002  point: comment=(t1==0 && t2==1) => 1 hier=top.t
-000002  point: comment=(t1==1 && t3==1) => 0 hier=top.t
-000004  point: comment=(t1==1 && t4==0) => 0 hier=top.t
-000002  point: comment=(t2==0 && t3==1) => 0 hier=top.t
-000003  point: comment=(t2==0 && t4==0) => 0 hier=top.t
-000002  point: comment=(t3==0 && t4==1) => 1 hier=top.t
```