Функциональная верификация цифровой аппаратуры

Лабораторная работа. Базовый тестовый стенд

October 28, 2025

Программный симулятор SystemVerilog

Verilator (https://verilator.org).

Интерфейс командной строки

```
$ verilator -Wno-lint --binary --vpi \
        <файл с исходным кодом DUT> \
        <файл с исходным кодом тестового стенда> \
        <... файлы с исходными кодами> \
        -o <имя результирующего исполняемого файла тестового стенда>
```

Используемые ключи запуска

- `--Wno-lint` Отключение линтера;
- `--binary` формирование исполняемого файла тестового стенда;

Результирующий исполняемый файл тестового стенда при успешной компиляции проекта появится в директории `obj_dir`.

Verilog

Интерактивный учебник-задачник по Verilog

`https://hdlbits.01xz.net/wiki/Main_Page`

Версия verilator в локальной сети

```
/auto/vgr/usr/x64/bin/verilator
```

Обёртка VCS в локальной (закрытой) сети

```
/auto/vgr/users/opium/lections/tools/vcswrap
```

Вещественная арифметика с плавающей точкой (IEEE-754 floating point arithmetic)

Тестируемый проект

Открытая реализация схем вещественной арифметики с плавающей точкой (IEEE-754 floating point arithmetic)

https://github.com/dawsonjon/fpu/tree/master

Интерфейс тестируемого устройства

```
module multiplier(
// входы
        input_a,        // значение первого аргумента
        input_b,        // значение второго аргумента
        input_a_stb,    // значимость первого аргумента
        input_b_stb,    // значимость второго аргумента
        output_z_ack,   // подтверждение приёма результат умножения
// служебные сигналы
        clk,            // синхросигнал
        rst,            // сброс
// выходы
        output_z,       // значение результата
        output_z_stb,   // признак значимости результата
        input_a_ack,    // подтверждение приёма первого аргумента
        input_b_ack);   // подтверждение приёма второго аргумента
```

Реализация тестового стенда

Элементы тестового стенда

SystemVerilog-часть

1. Модуль верхнего уровня;
2. экземпляр DUT;
3. синхросигнал и сигнал сброса;
4. формирование входных воздействий (значений операндов и сигналов готовности);
5. ожидание готовности результата;
6. сравнение результата с эталоном.

C++ -часть

- Импортируемая DPI-функция вычисления эталонного результата.

Режимы тестирования

В тестовом стенде следует реализовать три режима работы, определяющие источник значений входных данных.

Источник входных данных

- командная строка, разбор аргументов в тестовом стенде с помощью плюс-аргументов `$value$plusargs`: `+arg0=`, `+arg1=`;
- файл, имя которого передаётся в тестовый стенд через командную строку ключом `+from_file=`: каждая строка – набор значений аргументов для одного вычисления с разделителями между значениями;
- генератор псевдослучайных чисел с равномерным распределением `urandom`, режим ГПСЧ включается опцией `+random_mode=%d`, где численный параметр определяет число вычислений.

Зерно ГПСЧ задаётся из командной строки стандартным ключом симулятора:

- verilator: `+verilator+seed+<value>`
- vcs: `+ntb_random_seed=<value>`

Трассировка тестирования

Тестовый стенд должен

- для каждого тестового набора входных данных печатать в поток вывода симулятора входные данные и ожидаемый результат из DPI-части в нижеописанном формате.

  Формат вывода (в одну строку)

```
<dec_e (arg0)> (<hex (arg0)>) <op> <dec_e (arg1)> (<hex (arg1)>) = <dec_e (et_res)> (hex (et_res))

<op> (<dec (arg0)> (<hex (arg0)>)) = <dec_e (et_res)> (hex (et_res))

<op> (<dec_e (arg0)> (<hex (arg0)>)) = <dec (et_res)> (hex (et_res))
```

- для каждого тестового набора входных данных печатать входные данные, фактический результат и ожидаемый (эталонный) результат вычислений из SV-части тестового стенда нижеописанном формате.

  Формат вывода (в одну строку)

```
<dec_e (arg0)> (<hex (arg0)>) <op> <dec_e (arg1)> (<hex (arg1)>) = \
    <dec_e (et_res)> (hex (et_res)) VS <dec_e (res)> (hex (res))

<op> (<dec_e (arg0)> (<hex (arg0)>)) = <dec (et_res)> (hex (et_res)) VS <dec (res)> (hex (res))

<op> (<dec (arg0)> (<hex (arg0)>)) = <dec_e (et_res)> (hex (et_res)) VS <dec_e (res)> (hex (res))
```

Легенда формата вывода

- `arg0`, `arg1` – входные данные, операнды;
- `et_res`, `res` – эталонный и фактический результаты вычисления соответственно;
- `op` – символ операции, реализуемой тестируемым устройством;
- `dec (a)` – запись целого числа в десятичной системе счисления;
- `dec_e (a)` – запись вещественного числа *a* в десятичной экспоненциальной нотации;
- `hex` – запись числа *a* в шестнадатеричном представлении с соотвествующим основанию системы счисления префиксом.