

Языки описания схем

mk.cs.msu.ru → Лекционные курсы → Языки описания схем

Блок 12

Verilog:
Несхемный «Hello, World!»

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2023/2024, осенний семестр

Текстовый файл main.v

```
module main();  
    initial $display("Hello, World!");  
endmodule
```

Сборка и запуск в консоли Linux:

```
>ls  
main.v  
>iverilog main.v  
>ls  
main.v a.out  
>./a.out  
Hello, World!  
>
```

iverilog

```
>ls
main.v
>iverilog main.v
>ls
main.v a.out
>./a.out
Hello, World!
>
```

iverilog — это компилятор для \mathcal{V} , схожий по использованию с компилятором gcc для C/C++.

Результат сборки по умолчанию называется a.out, скрипт, который

- ▶ в Linux — исполняемый, а
- ▶ в Windows запускается программой vvp, обычно идущей в комплекте с iverilog

Модули

```
module main();  
    initial $display("Hello, World!");  
endmodule
```

Код \mathcal{V} состоит из модулей

Модуль можно понимать как описание подсхемы, или как аналог *класса/функции/функтора* в C/C++

Модуль состоит из

- ▶ **имени** (~ имя класса/функции)
- ▶ **портов** (~ аргументы функции)
- ▶ **тела** (~ тело функции/описание класса)

Экземпляры модулей (~ объекты класса) можно вставлять в другие модули

Как правило, **главным** (~ функция `main`) считается модуль, не имеющий ни одного экземпляра, независимо от его имени

Главных модулей может быть и несколько

Модули

```
module main();  
    initial $display("Hello, World!");  
endmodule
```

Объявление модуля:

```
module <имя>( <объявление портов> );  
    <тело>  
endmodule
```

Другой способ объявления модуля:

```
module <имя>( <имена портов через запятую> );  
    <дообъявление портов>  
    <тело>  
endmodule
```

Синтезируемость кода

```
module main();  
    initial $display("Hello, World!");  
endmodule
```

Все синтаксические конструкции языка делятся на три класса:

- ▶ **Поддерживаемые (синтезируемые)** имеют аппаратную семантику, могут служить полноценным описанием схемы
- ▶ **Игнорируемые** могут содержаться в описании схемы, но игнорируются (ничему не отвечают) в аппаратной семантике
- ▶ **Неподдерживаемые (несинтезируемые)** не имеют аппаратной семантики, их запрещено использовать в описании схемы

Далее по умолчанию полагается (если не сказано обратного), что все обсуждаемые конструкции поддерживаемы

Симуляция кода

```
module main();  
  initial $display("Hello, World!");  
endmodule
```

Выполнение кода \mathcal{V} в программной семантике далее будем называть программной **симуляцией**, и результат сборки кода в программной семантике для симуляции — программной **моделью**

Симуляция модели представляет собой выполнение действий в контексте **модельного времени**: особой переменной, в начале симуляции имеющей значение 0 и время от времени увеличивающей своё значение

Значение модельного времени во время выполнения действия будем называть **регионом** этого действия

`initial, $display`

```
module main();  
    initial $display("Hello, World!");  
endmodule
```

```
initial <команда>
```

Это игнорируемая **начальная процедура**, означающая, что следует выполнить *команду* в начале симуляции в регионе 0¹

```
$display("format", args ...)
```

Это игнорируемая команда, аналогичная команде **printf** языка C/C++ с добавленным переводом строки

Таким образом, в теле модуля выше сказано вывести строку "Hello, World!" в консоль в начале симуляции

¹ Точнее, это процедура, запускающаяся в начале выполнения и при выполнении добавляющая в регион 0 срочное действие выполнения *команды*, но пока не будем переусложнять и вводить много страшных терминов

Заключение

```
module main();  
    initial $display("Hello, World!");  
endmodule
```

```
>ls  
main.v  
>iverilog main.v  
>ls  
main.v a.out  
>./a.out  
Hello, World!  
>
```