Лабораторный практикум

«Проектирование цифровых устройств с помощью Verilog HDL»

Лабораторная работа №1 Введение в Verilog HDL

1.1 Возникновение языков описания цифровой аппаратуры

Цифровые устройства — это устройства, предназначенные для приёма и обработки цифровых сигналов. Цифровыми называются сигналы, которые можно рассматривать в виде набора дискретных уровней. В цифровых сигналах информация кодируется в виде конкретного уровня напряжения. Как правило выделяется два уровня — логический «0» и логическая «1».

Цифровые устройства стремительно развиваются с момента изобретения электронной лампы, а затем транзистора. Со временем цифровые устройства стали компактнее, уменьшилось их энергопотребление, возрасла вычислительная мощность. Так же разительно возросла сложность их структуры.

Графические схемы, которые применялись для проектирования цифровых устройств на ранних этапах развития, уже не могли эффективно использоваться. Потребовался новый инструмент разработки, и таким инструментом стали языки описания аппаратной части цифровых устройств (Hardware Description Languages, HDL), которые описывали цифровые структуры формализованным языком, чем-то похожим на язык программирования.

Совершенно новый подход к описанию цифровых схем, реализованный в языках HDL, заключается в том, что с помощью их помощью можно описывать не только структуру, но и поведение цифрового устройства.Окончательная структура цифрового устройства получается путём обработки таких смешанных описаний специальной программой — синтезатором.

Такой подход существенно изменил процесс разработки цифровых устройств, превратив громоздкие, тяжело читаемые схемы в относительно простые и доступные описания поведения.

В данном курсе мы рассмотрим язык описания цифровой аппаратуры Verilog HDL — одни из наиболее распространённых на текущий момент. И начнём мы с разработки наиболее простых цифровых устройств — логических вентилей.

1.2 HDL описания логических вентилей

Логические вентили реализуют функции алгебры логики: И, ИЛИ, Исключающее ИЛИ, НЕ. Напомним их таблицы истинности:

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 1.1: И

a	b	$a \vert b$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 1.2: ИЛИ

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Таблица 1.3: Исключающее ИЛИ

a	$\bar{a}$
0	1
1	0

Таблица 1.4: НЕ

Начнём знакомиться с Verilog HDL с описания логического вентиля «И». Ниже приведен код, описывающий вентиль с точки зрения его структуры:

```
module and_gate(
        input  a,
        input  b,
        output result)

assign result = a & b;

endmodule
```

Листинг 1.1: Модуль, описывающий вентиль «И»

Описанный выше модуль можно представить как некоторый «ящик», в который входит 2 провода с названиями *«a»* и *«b»* и из которого выходит один провод с названием *«result»*. Внутри этого блока результат выполнения операции «И» (в синтаксисе Verilog записывается как «&») над входами соединяют с выходом.

Схемотично изобразим этот модуль:

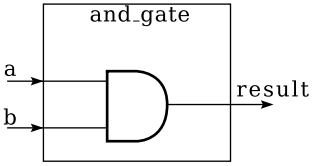

Рис. 1.1: Структура модуля «and_gate»

Аналогично опишем все оставшиеся вентили:

```
module or_gate(
        input  a,
        input  b,
        output result)

assign result = a | b;

endmodule
```

Листинг 1.2: Модуль, описывающий вентиль «ИЛИ»

```
module xor_gate(
        input  a,
        input  b,
        output result)

assign result = a ^ b;

endmodule
```

Листинг 1.3: Модуль, описывающий вентиль «Исключающее ИЛИ»

```
module not_gate(
        input  a,
        output result)

assign result = ~a;

endmodule
```

Листинг 1.4: Модуль, описывающий вентиль «НЕ»

В проектировании цифровых устройств логические вентили наиболее часто используются для формулировки и проверки сложных условий, например:

```
if ( (a & b) | (~c) ) begin
        ...
end
```

Листинг 1.5: Пример использования логических вентилей

Условие будет выполняться либо когда *не* выполнено условие *«c»*, либо когда одновременно выполняются условия *«a»* и *«b»*. *Здесь и далее под условием понимается логический сигнал, отражающий его истинность.*

В качестве входов, выходов и внутренних соединений в блоках могут использоваться шины — группы проводов. Ниже приведен пример работы с шинами:

```
module bus_or(
        input  [7:0] x,
        input  [7:0] y,
        output [7:0] result);

assign result = x | y;

endmodule
```

Листинг 1.6: Модуль, описывающий побитовое «ИЛИ» между двумя шинами

Это описание описывает побитовое «ИЛИ» между двумя шинами по 8 бит. То есть описываются восемь логических вентилей «ИЛИ», каждый из которых имеет на входе соответствующие разряды из шины «*x*» и шины «*y*».

При использовании шин можно в описании использовать конкретные биты шины и группы битов. Для этого используют квадратные скобки после имени шины:

```
module bitwise_ops(
        input  [7:0] x,
        output [4:0] a,
        output       b,
        output [2:0] c);

assign a = x[5:1];
assign b = x[5]   | x[7];
assign c = x[7:5] ^ x[2:0];

endmodule
```

Листинг 1.7: Модуль демонстрирующий битовую адресацию шин

Такому описанию соответствует следующая структурная схема:

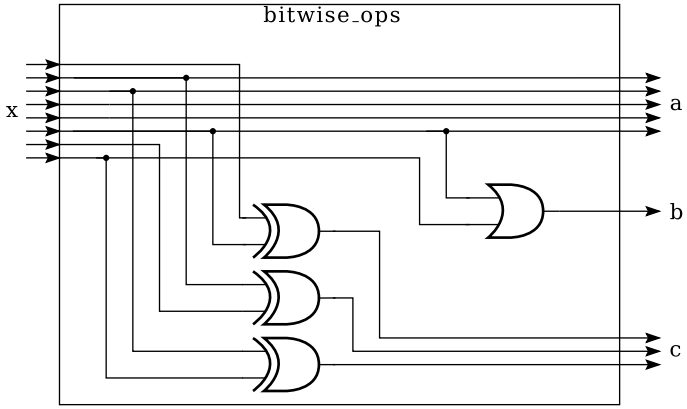

Рис. 1.2: Структура модуля «bitwise_ops»

Лабораторная работа №6
FLASH память

2.1 Возникновение языков описания цифровой аппаратуры