Лабораторная работа №1
Введение в Verilog . Логические вентили.

Изучаемый Вами курс называется «Программируемые логические интегральные схемы» (ПЛИС).

Программируемые логические интегральные схемы возникли как ответ на потребность разработчиков аппаратуры в структуре, реализующей логическую интегральную схему, функционал которой можно было бы изменять простым путём (программированием).

Логическая интегральная схема — это схема, предназначенная для приёма и обработки цифровых сигналов, то есть таких сигналов, состояние которых следует рассматривать в виде набора дискретных состояний или логических уровней. Такими состояниями являются логические «0», «1» и так называемое «Z»-состояние.

На базе логических интегральных схем выполняют проектирование цифровых устройств. Цифровые устройства стремительно развиваются с момента изобретения электронной лампы, а затем транзистора. Со временем цифровые устройства стали существенно меньше, уменьшилось их энергопотребление. Настолько же значительно возросла сложность их структуры и их функциональные возможности.

С ростом сложности цифровых устройств, графические схемы, которые применялись для проектирования, уже не могли эффективно использоваться. Потребовались новые инструменты разработки и ими стали языки описания аппаратуры (Hardware Description Language (HDL)), которые описывают структуру или поведение проектируемого устройства формализованным языком, схожим с языками программирования высокого уровня.

В этом курсе мы будем использовать один из наиболее распространённых языков описания цифровой аппаратуры Verilog. Этот язык поддерживает несколько уровней абстракции, обеспечивающий проектирование устройств различными способами:

1) Поведенческий уровень (Behavioral Level). Это самый высокий уровень абстракции, он представляет собой описание алгоритма функционирования устройства без привязки к аппаратной реализации. Проектирование на этом уровне подобно обычному программированию на других языках высокого уровня.
2) Уровень потока данных (Dataflow Level). На этом уровне описывается процесс обмена данными между регистрами и операции, выполняемые с этими данными. Данный уровень иногда называют уровнем регистровых пересылок (Register Transfer Level - RTL). При этом в описании используются только синтезируемые конструкции – то есть такие, которые можно преобразовать в аппаратную схему без потерь в функционале.
3) Вентильный уровень (Gate Level). Описание на этом уровне подобно графическому представлению проекта, т.е. состоит из описания логических вентилей и соединений между ними.
4) Транзисторный, или ключевой уровень (Switch Level). На этом самом низшем уровне абстракции описываются ключевые транзисторы и соединения между ними.

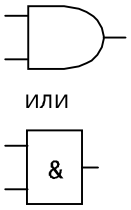
В процессе изучения этого курса Вы в основном будете работать на уровнях 3 и 2. Структура проектируемого устройства в таком случае получается в результате обработки Verilog-описания специальной программой — синтезатором. Этот подход существенно изменил процесс разработки цифровых устройств, превратив громоздкие, тяжело читаемые схемы в относительно простые и доступные описания поведения.

И начнём с разработки наиболее простых цифровых устройств — логических вентилей.

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

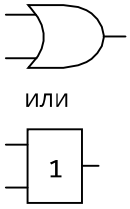
Операция «И» («AND», «∧», «·») – логическое умножение

x	y	$x \cdot y$
0	0	0
1	0	0
0	1	0
1	1	1



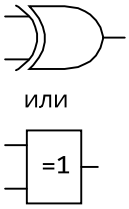
Операция «ИЛИ» («OR», «∨», «+») – логическое сложение

x	y	$x + y$
0	0	0
1	0	1
0	1	1
1	1	1



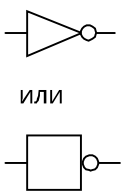
Операция «Исключающее ИЛИ» («XOR», «⊕», сложение по модулю 2)

x	y	$x \oplus y$
0	0	0
1	0	1
0	1	1
1	1	0



Операция «НЕ» («NOT», «$\bar{x}$»)

x	$\bar{x}$
0	1
1	0



Теперь опишем логические вентили, реализующие эти функции на языке Verilog.

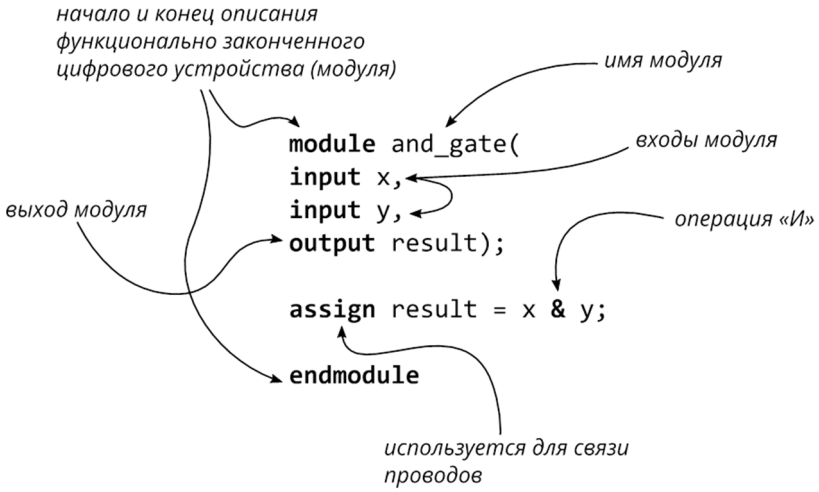
Здесь и далее в коде описания цифровых устройств жирным шрифтом отмечены ключевые слова и символы языка Verilog.

Основой языковых конструкций языка Verilog является модульная структура, когда разрабатываемое устройство состоит из модулей, связанных между собой с соблюдением иерархии – т.е. модули верхнего и нижнего уровней. Каждый из этих модулей, представляет собой устройство, выполняющее определённую, уникальную для разрабатываемого устройства функцию.

Модуль должен иметь входы и выходы, обеспечивающие получение и передачу информации во внешние устройства, причём такими устройствами могут быть другие модули. Verilog – описание разрабатываемого модуля должно быть записано в файл с расширением « .v», причём название или имя модуля должно совпадать с названием файла для обеспечения работы синтезатора и облегчения навигации внутри проекта.

Так как Verilog-описание после синтеза представляет собой схему, то синтезируемыми являются следующие типы данных: **wire, tri, reg**. Тип **wire** – «провод» предназначен для описания соединений в схеме, разновидностями так же являются input и output – «вход» и выход». Кроме того, этот тип используется для создания комбинационных схем. Тип **tri** аналогичен типу wire, но допускает подключение такой цепи к нескольким источникам сигнала. Тип **reg** (register) – «регистр» чаще всего используется при работе с аппаратными регистрами, состоящих из триггеров. В этом случае результат синтеза будет определяться использованием переменной этого типа, например: сдвиговый регистр или счётчик. Перечисленные типы данных могут объявляться как векторы, т.е. быть многоразрядными. Если при объявлении разрядность не указана, то по умолчанию переменная является однобитной.

Начнём с описания логического вентиля, выполняющего операцию «И»:



Описанный выше модуль можно представить как некоторый «ящик», в который входит 2 провода с названиями ***x*** и ***y*** 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.

Ключевое слово **assign** используется для выполнения непрерывного присваивания, результатом синтеза такой конструкции является комбинационная схема.

Аналогично опишем все остальные логические вентили.

Вентиль «ИЛИ»:

```
module or_gate(
input x,
input y,
output result);

assign result = x | y;

endmodule
```

Вентиль «Исключающее ИЛИ»:

```
module xor_gate(
input x,
input y,
output result);

assign result = x ^ y;

endmodule
```

Вентиль «НЕ»:

```
module not_gate(
input x,
output result);

assign result = ~x;

endmodule
```

В поведенческом описании цифровых устройств логические вентили наиболее часто используются для формулирования и проверки сложных условий, например:

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

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.

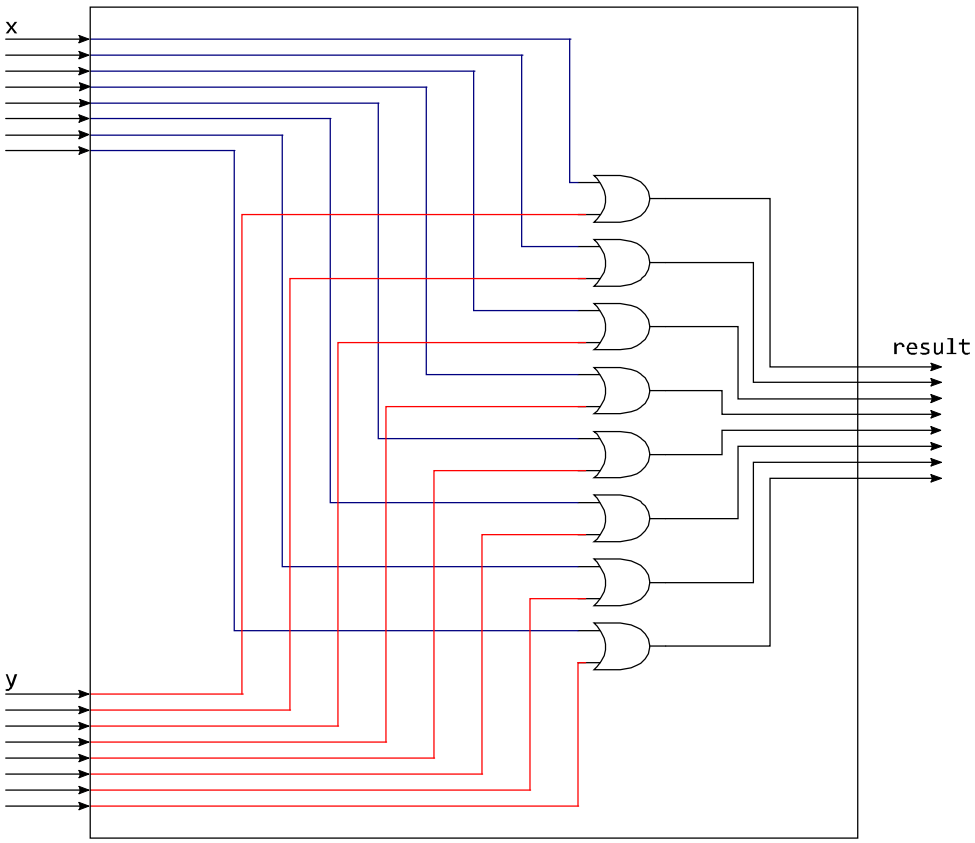
```
module or_bus(
input [7:0] x,
input [7:0] y,
output [7:0] result);

assign result = x | y;

endmodule
```

Это описание реализует 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**.

Проиллюстрируем это описание схемой, ему соответствующей:



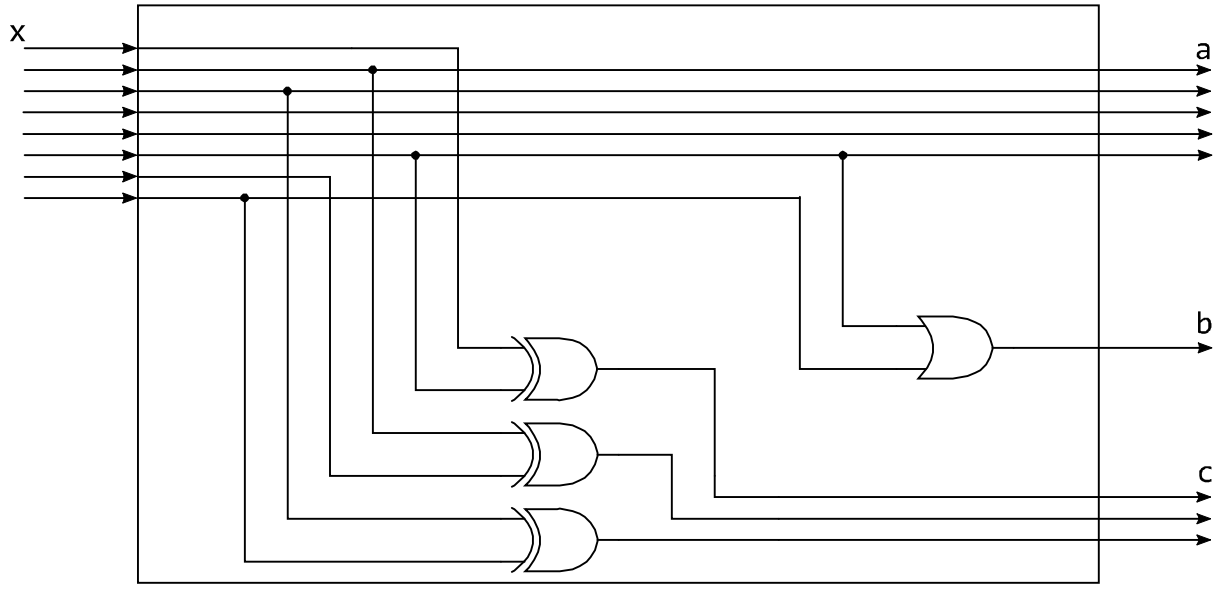
При использовании шин можно в описании использовать конкретные биты шины и группы битов. Для этого используют квадратные скобки после имени шины:

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

assign a = x[6:2];
assign b = x[5] | x[7];
assign c = x[7:5] ^ x[2:0];

endmodule
```

Такому описанию соответствует следующая схема:



Задание для выполнения лабораторной работы:

1. Описать на языке Verilog функцию алгебры логики (ФАЛ), заданную в виде формулы, построить временные диаграммы для проверки его работы помощью САПР Altera Quartus II.

Задание является индивидуальным. Если в задании встречаются переменные с цифровыми индексами, их следует описывать в виде шин.

2. Привязать входы модуля к переключателям SW, отладочной платы, а выход к диодам LEDG (таким образом, чтобы на все диоды подавалось одинаковое значение). Получить файл конфигурации для ПЛИС и продемонстрировать работу этого модуля.
3. Сформулированное на неформальном языке логическое выражение описать в виде ФАЛ на языке Verilog.

Пример индивидуального задания:

1. $f(a_0, a_1, a_2, s, m) = a_0 \cdot a_1 \cdot m + a_2 \cdot s \cdot m \oplus \bar{s} \cdot a_1$
2. Чтобы можно было загорать, должно быть солнечно и тепло

Решение индивидуального задания:

```
module lab_1_1(
input [2:0] a,
input s,
input m,
output f);

assign f = (a[0] & a[1] & m) | ((a2 & s & m) ^ (s & ~a[1]));

endmodule
```

```
module lab_1_2(
input sun_is_there,
input it_is_warm,
output you_can_sunbath);

assign you_can_sunbath = sun_is_there & it_is_warm;

endmodule
```

Базовые вопросы к защите лабораторной работы:

1. Какие строки кода вашей работы являются описанием входов и выходов вашего цифрового устройства?
2. Какие строки отвечают за структуру модуля?
3. По памяти выпишите таблицы истинности для логических функций «И», «ИЛИ», «XOR».
4. Для каких целей в описаниях цифровых устройств наиболее часто применяются логические вентили?