

# ПАРАМЕТРИЗОВАННЫЕ МОДУЛИ

Лекционное занятие



# Параметризованные модули

Параметризация в **SystemVerilog** позволяет создавать модули, которые могут принимать параметры для конфигурации их поведения и структуры. Это делает код более гибким и повторно используемым.

Параметры могут быть использованы для определения размеров шин, количества регистров, задержек и других аспектов модуля.

Параметры объявляются внутри модуля с использованием ключевого слова **parameter**. Также можно использовать **localparam** для параметров, которые не могут быть изменены извне

При **инстанцировании** экземпляра модуля параметры могут быть переопределены, чтобы конфигурировать экземпляр модуля.

Инстанцирование модулей в **System Verilog** — это процесс создания экземпляров повторно используемых модулей в рамках более крупного проекта .

Он допускает иерархический дизайн, в котором сложные системы могут быть построены путем объединения более простых модулей. Этот модульный подход улучшает читаемость, масштабируемость и удобство обслуживания проекта **System Verilog**

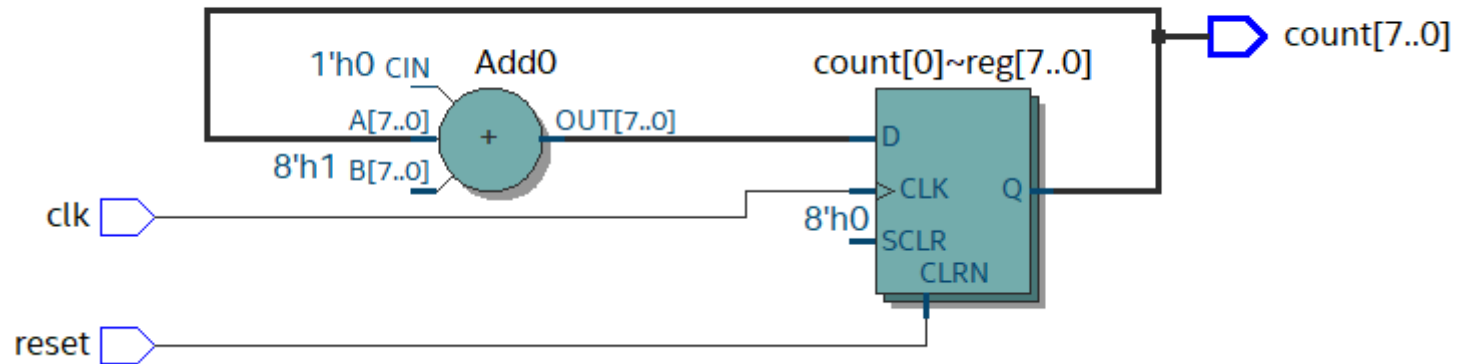
# Пример параметризованного модуля

```
module counter #(parameter WIDTH = 8) (  
    input logic clk,  
    input logic reset,  
    output logic [WIDTH-1:0] count);  
    always_ff @(posedge clk or posedge  
reset) begin  
        if (reset)  
            count <= 0;  
        else  
            count <= count + 1;  
    end  
endmodule
```

В этом модуле параметр **WIDTH** задает ширину счетчика. По умолчанию он равен 8, но может быть переопределен при инстанцировании.

---

# Пример параметризованного модуля



Синтезированная схема параметризованного модуля

# Параметры и другие возможности

В **SystemVerilog** параметры позволяют модулям быть более гибкими и настраиваемыми. Помимо простых числовых параметров, в **SystemVerilog** также поддерживаются типизированные параметры, многопараметричные модули и локальные параметры.

Используя типизированные параметры, многопараметричные модули и локальные параметры, вы можете создавать сложные и адаптивные системы, которые легко конфигурируются под конкретные задачи.

# Типизированные параметры

```
module fifo #(parameter type DATA_T = logic
[7:0]) (
    input logic clk,
    input logic reset,
    input DATA_T data_in,
    output DATA_T data_out);
DATA_T fifo_mem [0:15]; // Пример FIFO
// памяти с типом данных DATA_T
...
endmodule
```

ТИПИЗИРОВАННЫЕ ПАРАМЕТРЫ

Типизированные параметры позволяют передавать типы данных в качестве параметров, что делает модуль более гибким. Это особенно полезно, когда вы хотите создать модуль, который может работать с различными типами данных.

---

# Многопараметричные модули

```
module alu #(parameter WIDTH = 32,  SIGNED_MODE = 0)
(
    input logic [WIDTH-1:0] a,
    input logic [WIDTH-1:0] b,
    input logic [2:0] op,
    output logic [WIDTH-1:0] result);
always_comb begin
    case (op)
        3'b000: result = a + b; // Пример операции сложения
        3'b001: result = a - b; // Пример операции вычитания
        default: result = {WIDTH{1'b0}};
    endcase
end
endmodule
```

МНОГОПАРАМЕТРИЧНЫЕ МОДУЛИ



# Многопараметричные модули

- Модули могут иметь несколько параметров, что позволяет более точно настраивать их функциональность и поведение.
- В приведенном примере модуля `alu` имеются два параметра: `WIDTH` для задания ширины операндов результата, и `SIGNED_MODE` для указания является ли операция знаковой.

# Локальные параметры

```
module some_module #(parameter SIZE = 8)
(
    input logic [SIZE-1:0] data);

localparam HALF_SIZE = SIZE / 2;
logic [HALF_SIZE-1:0] half_data;

always_comb begin
    half_data = data[HALF_SIZE-1:0];
end
endmodule
```

ЛОКАЛЬНЫЕ ПАРАМЕТРЫ

`localparam` используется для определения параметров, которые не могут быть изменены извне модуля. Это удобно для внутренних констант, которые зависят от других параметров.

---

# Преимущества использования параметризованных модулей

- Гибкость дизайна: Возможность переопределения параметров при инстанцировании модулей позволяет легко адаптировать их к различным требованиям проекта.
- Повторное использование кода: Один и тот же модуль может быть использован в различных частях проекта с различными настройками, что сокращает время разработки и уменьшает количество ошибок.
- Упрощенное обслуживание и расширение: Параметризованные модули упрощают добавление новых функций и улучшений, так как изменения могут быть легко внесены через параметры.
- Типизированные параметры и многопараметричные модули: Эти возможности обеспечивают более высокую степень абстракции и гибкости, позволяя использовать модули с различными типами данных и конфигурациями.
- Локальные параметры: Использование `localparam` для внутренних констант повышает читаемость и надежность кода.

# Параметризованные N-битные двухвходовые мультиплексоры

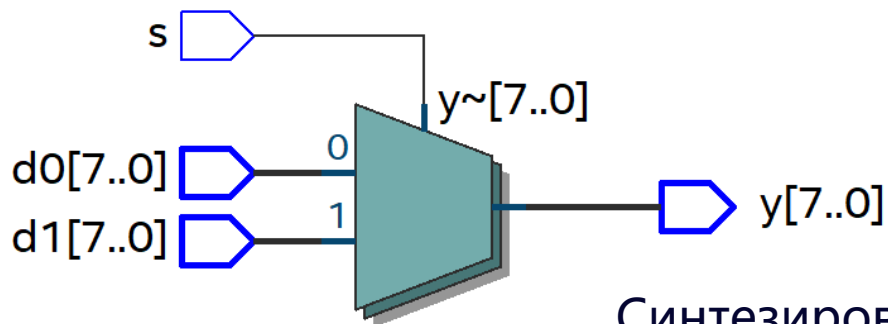
В следующем коде объявляется параметризованный двухвходовой мультиплексор с шириной входов, равной по умолчанию восьми битам, который затем используется для создания четырехвходовых мультиплексоров с восьмибитными и двенадцатибитными входами.

# Двухвходовый мультиплексор

```
module mux2 #(parameter width = 8)
(
    input logic [width-1:0] d0, d1,
    input logic s,
    output logic [width-1:0] y);

    assign y = s ? d1 : d0;
endmodule
```

ДВУХВХОДОВЫЙ МУЛЬТИПЛЕКСОР



Синтезированная схема  
двухвходового мультиплексора

В SystemVerilog возможна конструкция `#(parameter ...)` перед списком входов и выходов для определения параметров модуля. Оператор `parameter` состоит из параметра по имени `width` со значением по умолчанию, равным 8.

# 8-битный четырехходовый мультиплексор

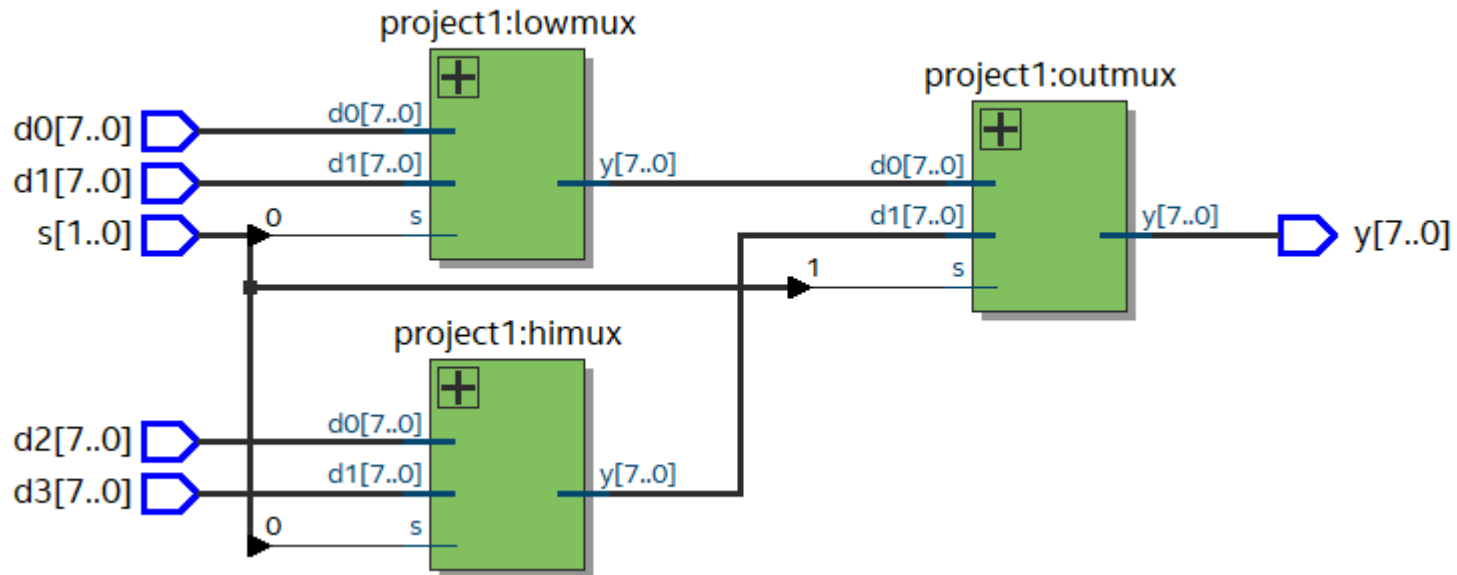
```
module mux4_8(  
    input logic [7:0] d0, d1, d2, d3,  
    input logic [1:0] s,  
    output logic [7:0] y);  
  
    logic [7:0] low, hi;  
    mux2 lowmux(d0, d1, s[0], low);  
    mux2 himux (d2, d3, s[0], hi);  
    mux2 outmux(low, hi, s[1], y);  
endmodule
```

ЧЕТЫРЕХВОДНОЙ МУЛЬТИПЛЕКСОР

8-битный четырехходовой мультиплексор состоит из трех экземпляров двухходового мультиплексора с шириной входов, установленной по умолчанию.

---

# 8-битный четырехвходной мультиплексор



Синтезированная схема 8-битного четырехвходного мультиплексора

# 12-битный четырехвходной мультиплексор

```
module mux4_12 (  
    input logic [11:0] d0, d1, d2, d3,  
    input logic [1:0] s,  
    output logic [11:0] y);  
  
    logic [11:0] low, hi;  
    mux2 #(12) lowmux (d0, d1, s[0], low);  
    mux2 #(12) himux  (d2, d3, s[0], hi );  
    mux2 #(12) outmux (low, hi, s[1], y );  
endmodule
```

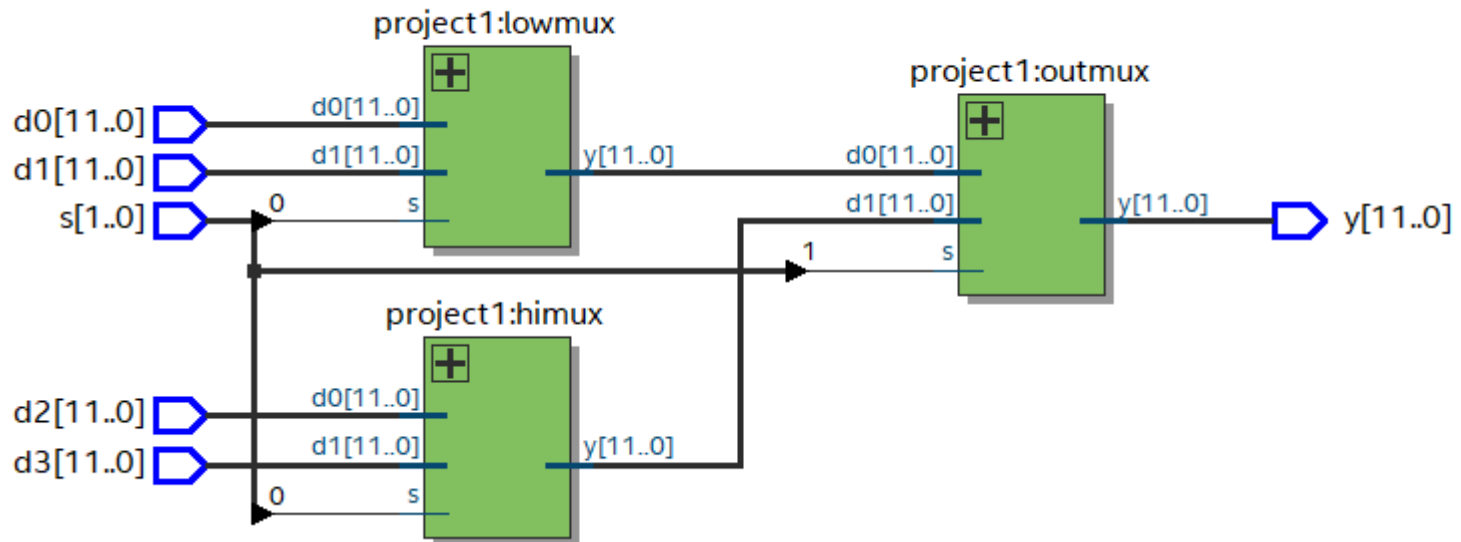
ЧЕТЫРЕХВХОДНОЙ МУЛЬТИПЛЕКСОР

В 12-битном четырехвходовом мультиплексоре `mux4_12` понадобится переопределить ширину входов с помощью конструкции `#( )` перед именем экземпляра (`instance`).

---



# 12-битный четырехвходной мультиплексор



Синтезированная схема 12-битного четырехвходного мультиплексора

# Литература

Дэвид М. Хэррис, Сара Л. Хэррис Цифровая схемотехника  
и архитектура компьютера. Второе издание. Morgan Kaufman, 2013 г .  
– 1648 стр.