

# Языки описания схем

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

## Блок 14

Verilog:  
ещё пара слов о процедурных командах

Лектор:  
**Подымов Владислав Васильевич**

E-mail:  
**valdus@yandex.ru**

# Вступление

```
initial <команда>  
always <команда>
```

В *блоке 12* обсуждались основные команды, позволяющие управлять значениями переменных:

- ▶ Составная команда: `begin <последовательность команд> end`
- ▶ Блокирующее присваивание: `x = E;`
- ▶ Неблокирующее присваивание: `x <= E;`

В процедурах можно использовать и другие *полезные* команды

## ∪: пустая команда; ветвление

Пустая команда:

```
;
```

Выполнение команды: ничего не происходит

Ветвление:

```
if(<выражение>) <команда> else <команда>
```

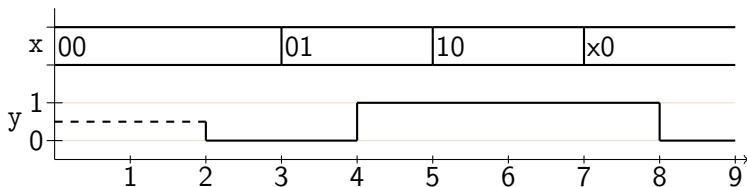
Выполнение команды:

- ▶ Вычисляется значение val <выражения>
- ▶ Если val совпадает с (00...0) или содержит разряды x, z, то выполняется правая <команда>
- ▶ Иначе выполняется левая <команда>

```
if(<выражение>) <команда> = if(<выражение>) <команда> else ;
```

## ∪: пустая команда; ветвление

```
module test;  
  reg [1:0] x;  
  reg y;  
  always #2 if(x) y = 1; else y = 0;  
  initial begin  
    x = 0;  
    #3 x = 2'b1;  
    #2 x = 2'b10;  
    #2 x = 2'bx0;  
    #2 $finish;  
  end  
endmodule
```



## У: команды выбора

Точный выбор:

```
case (<выражение>)  
  <случай>  
  <случай>  
  ...  
  <случай>  
  default : <команда>  
endcase
```

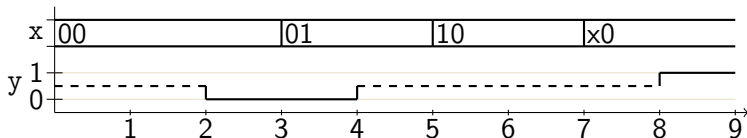
<случай> ::= <выражение> : <команда>

Выполнение команды:

- ▶ Вычисляется значение val <выражения> в скобках
- ▶ Последовательно по порядку до **успеха** обрабатываются <случай>:
  - ▶ Вычисляется значение sval <выражения> <случая>
  - ▶ Если sval поразрядно совпадает с val,  
то **успех**, и выполняется <команда> случая
- ▶ Если все случаи обработаны неуспешно,  
то выполняется <команда> после слова “default”

## V: команды выбора

```
module test;
  reg [1:0] x; reg y;
  always #2
    case(x)
      2'b00: y = 0;
      2'bx0: y = 1;
      default: y = 1'bx;
    endcase
  initial begin
    x = 0;
    #3 x = 2'b1;
    #2 x = 2'b10;
    #2 x = 2'bx0;
    #2 $finish;
  end
endmodule
```



## У: команды выбора

Выбор по шаблону:

```
casex (<выражение>)  
  <случай>  
  <случай>  
  ...  
  <случай>  
  default : <команда>  
endcase
```

Выполнение команды отличается от выполнения case только критерием успеха

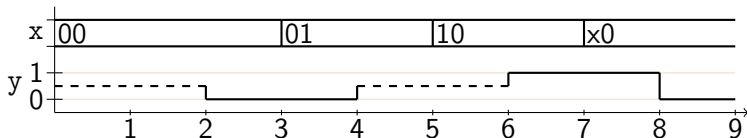
**Успех** ⇔

для каждой пары соответствующих разрядов значений val, sval выражения в скобках и выражения случая верно следующее:

**если значения обоих разрядов булевы, то эти значения совпадают**

## V: команды выбора

```
module test;
  reg [1:0] x; reg y;
  always #2
    casex(x)
      2'b00: y = 0;
      2'bx0: y = 1;
      default: y = 1'bx;
    endcase
  initial begin
    x = 0;
    #3 x = 2'b1;
    #2 x = 2'b10;
    #2 x = 2'bx0;
    #2 $finish;
  end
endmodule
```





## У: команды выбора

### Другие возможности команд выбора

Строку “default : <команда>” в командах выбора можно не писать:  
отсутствие “default” = default : ;

Выражения случаев не **обязаны** быть константами

Слева от двоеточия в <случаях>  
можно записывать **несколько** выражений через запятую:  
подразумевается несколько случаев с одинаковой командой

## У: поддержка значения $\mathcal{X}$ в процедурах

С точки зрения аппаратной семантики, **явное** использование значения  $\mathcal{X}$  в коде *поддерживается* только в следующих случаях:

1. В правой части присваиваний, если к этому значению не применяются никакие операции

**Можно**

```
x <= 2'bx0;  
assign x = 2'bx0;
```

**Нельзя**

```
x <= 2'bx0 + 1;  
assign x = 2'bx0 + 1;
```

2. В выражениях случаев casex

**Можно**

```
casex(x)  
  2'b00: y = 0;  
  2'bx0: y = 1;  
  default: y = 1'bx;  
endcase
```

**Нельзя**

```
case(x)  
  2'b00: y = 0;  
  2'bx0: y = 1;  
  default: y = 1'bx;  
endcase
```

Все другие виды **явного** использования значения  $\mathcal{X}$  являются *неподдерживаемыми*