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

ВМК МГУ, 2020–2021, осенние семестры

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

$\mathcal{V}$: пустая команда; ветвление

Пустая команда:

```
;
```

Выполнение команды: ничего не происходит

Ветвление:

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

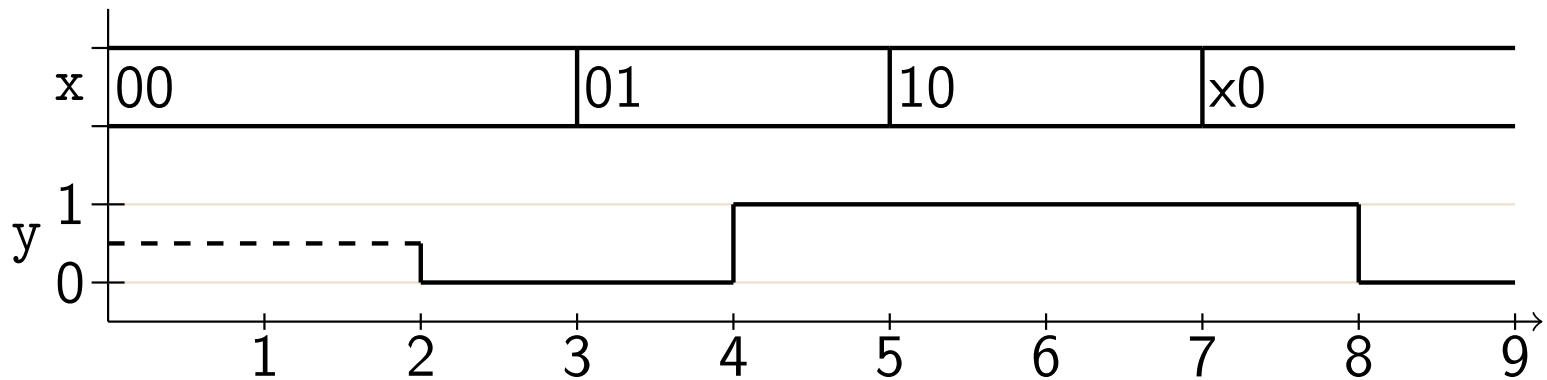
Выполнение команды:

- 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 ;`

$\mathcal{V}$: 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
```



$\mathcal{V}$: команды выбора

Точный выбор:

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

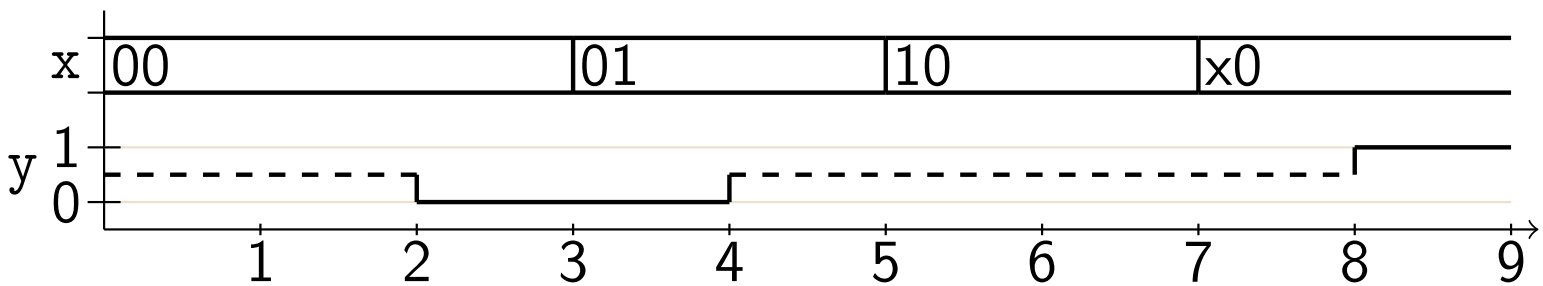
```
<случай> ::= <выражение> : <команда>
```

Выполнение команды:

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

$\mathcal{V}$: 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
```



$\mathcal{V}$: команды выбора

Выбор по шаблону:

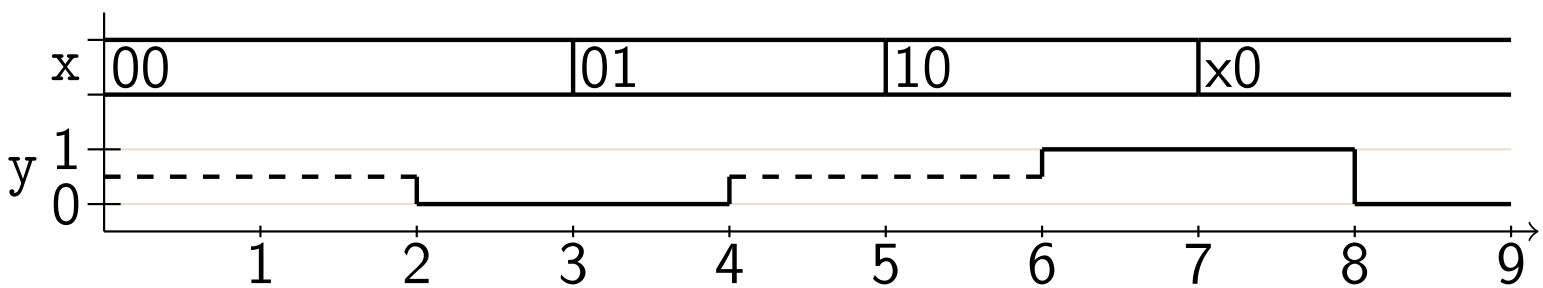
```
casex(<выражение>)
  <случай>
  <случай>
  ...
  <случай>
  default : <команда>
endcase
```

Выполнение команды отличается от выполнения case только критерием успеха

Успех $\Leftrightarrow$
для каждой пары соответствующих разрядов значений val, cval выражения в скобках и выражения случая верно следующее:
если значения обоих разрядов булевы, то эти значения совпадают

$\mathcal{V}$: 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
```



$\mathcal{V}$: 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

$\mathcal{V}$: 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	Нельзя
`x <= 2'bx0;`	`x <= 2'bx0 + 1;`
`assign x = 2'bx0;`	`assign x = 2'bx0 + 1;`

2. В выражениях случаев `casex`

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