Оператори за присвояване

Съществуват два основни типа присвояване:

— **непрекъснато** (continuous) присвояване, което дава стойности на вериги или променливи

— **процедурно** (procedural) просвояване, което дава стойност на променливи

**Непрекъснати просвоявания**

Непрекъснатото присвояване е подобно на **комбинационна схема**, чиито изход е свързан с верига или променлива.

Обектът от лявата страна на оператора за присвояване се променя когато се промени стойността на израза от дясната страна на оператора.

 logic carry\_out, carry\_in;

 logic [3:0] sum\_out, ina, inb;

 assign {carry\_out, sum\_out} = ina + inb + carry\_in;

**Процедурни присвоявания**

Процедурното присвояване дава стойност на **променлива**, като това действие протича “мигновенно”. Променливата съхранява присвоената й стойност до следващото процедурно просвояване.

Процедурните присвоявания се срещат в процедури, като например always, initial, task и function.

**Блокиращи Процедурни присвоявания**

Блокиращо процедурно присвояване трябва да бъде изпълнено **преди следващите го оператори**.

always @ (posedge clk) begin

 q1\_tmp = d;

 q1 = q1\_tmp;

 end

**Неблокиращи Процедурни присвоявания**

Неблокиращите процедурни присвоявания не променят веднага стойността на променливата от лявата страна на оператора.

Те определят новата стойност и **планират промяната**, която ще се извърши на края на т.нар. nonblocking assign update event region.

**Пример - блокиращи и неблокиращи присвоявания**

|  |  |
| --- | --- |
| module blocking\_vs\_nonblocking( output logic q1,q2, input clk, d); logic q1\_tmp, q2\_tmp; always @ (posedge clk) begin q1\_tmp = d; q1 = q1\_tmp;  end always @ (posedge clk) begin q2\_tmp <= d; q2 <= q2\_tmp; endendmodule |  |

**Пример - блокиращи и неблокиращи присвоявания**

`timescale 1ns / 1ps

 logic a, b, c, d, e, f;

 // блокиращи

 initial begin

 a = #10 1; // a ще стане 1 на 10 **единици време** от началото на симулацията

 b = #2 0; // b ще стане 0 на 12 е.в. от началото на симулацията

 c = #4 1; // c ще стане 1 на 16 е.в. от началото на симулацията

 end

 // неблокиращи

 initial begin

 d <= #10 1; // d ще стане 1 на 10 е.в. от началото на симулацията

 e <= #2 0; // e ще стане 0 на 2 е.в. от началото на симулацията

 f <= #4 1; // f ще стане 1 на 4 е.в. от началото на симулацията

 end



##

|  |  |
| --- | --- |
| Пример - blocking assignmentmodule blocking\_assignment(input clk, sin, output logic[3:0] qout);  always\_ff @(posedge clk) begin qout[0] = sin; qout[1] = qout[0]; qout[2] = qout[1]; qout[3] = qout[2]; end;endmoduletime=0 qout=xxxxtime=150 qout=1111time=650 qout=0000 | Пример - non-blocking assignmentmodule non\_blocking\_assignment(input clk, sin, output logic[3:0] qout);  always\_ff @(posedge clk) begin qout[0] <= sin; qout[1] <= qout[0]; qout[2] <= qout[1]; qout[3] <= qout[2]; end;endmoduletime=0 qout=xxxxtime=150 qout=xxx1time=250 qout=xx11time=350 qout=x111time=450 qout=1111time=650 qout=1110time=750 qout=1100time=850 qout=1000time=950 qout=0000 |

blocking\_assignment





non\_blocking\_assignment





|  |  |
| --- | --- |
| **Как се променят стойностите на а и b?**module two\_step; logic a = 0; logic b = 1; logic clk = 0; always clk = #5 ~clk; always\_ff @(posedge clk) begin a **<=** b; b **<=** a; end initial $monitor("%t %b %b",$time,a,b); initial #50 $finish;endmodule |  a b 0 0 1 50 1 0 150 ? ? 250 ? ? 350 ? ? 450 ? ? |