Процедури - always и initial

**Структура на SystemVerilog модел**

Основната структурна единица на SV моделите е **module**. Модулите могат да съдържат йерархия от други модули, проводници (wires), **task** и **function** декларации, както и процедурни оператори в **always** и **initial** блокове.

## Initial процедури

Initial процедурите се изпълняват само веднъж. Използват се при разработване на тестове. Те се игнорират при синтеза.

Пример - установяване на начални стойности при симулация

initial begin

a = 0; // initialize a

for (int index = 0; index < size; index++)

memory[index] = 0; // initialize memory word

end

Пример - описание на тестови въздействия

initial begin

inputs = 'b000000; // initialize at time zero

#10 inputs = 'b011001; // first pattern

#10 inputs = 'b011011; // second pattern

#10 inputs = 'b011000; // third pattern

#10 inputs = 'b001000; // last pattern

end

Пример - тест

module shift\_register\_test\_1;

localparam N = 6;

logic clock, reset\_n, load;

logic [N-1:0] data\_in, data\_out;

shift\_register #(.N(N)) uut (.\*);

initial begin

clock = 0;

forever #50 clock = ~clock;

end

initial begin

reset\_n = 0;

data\_in = 1;

load = 0;

#100 reset\_n = 1;

#20 load = 1;

#100 load = 0;

#5000 $finish;

end

initial begin

$monitor("%b %b %b",reset\_n, load, data\_out);

$dumpfile("dump.vcd");

$dumpvars;

end

endmodule

## Always процедури

Използват се за описание на повтарящи се процеси. Изпълняват се непрекъснато докато продължава симулацията. Съществуват четири варианта: always, always\_comb, always\_latch, and always\_ff.

### always

always @(sensitivity list) <begin> <procedural statements> <end>

Това е always процедура с общо предназначение. С подходящи механизми за активиране (timing controls) тя може да описва както комбинационни, така и последователни схеми.

### always\_comb

Използва се за моделиране на комбинационни схеми.

always\_comb

a = b & c;

always\_comb

d <= #1ns b & c;

Разлики от always процедурата

— няма явен sesitivity list

— променливите от лявата страна на оператори за присвояване не трябва да бъдат променяни от други процеси

— процедурата се изпълнява автоматично в началото на симулацията

tools should check and warn if the behavior does not represent combinational logic.

### always\_latch

Използва се за моделиране на схеми с тригери управлявани по ниво.

always\_latch

if(ck) q <= d;

software tools should perform additional checks and warn if the behavior in an always\_latch construct does not represent latched logic

## Пример - always

module always\_example(

input [3:0] a,b,

input mode,

output logic [3:0] o1, o2

);

always @(mode,a,b)

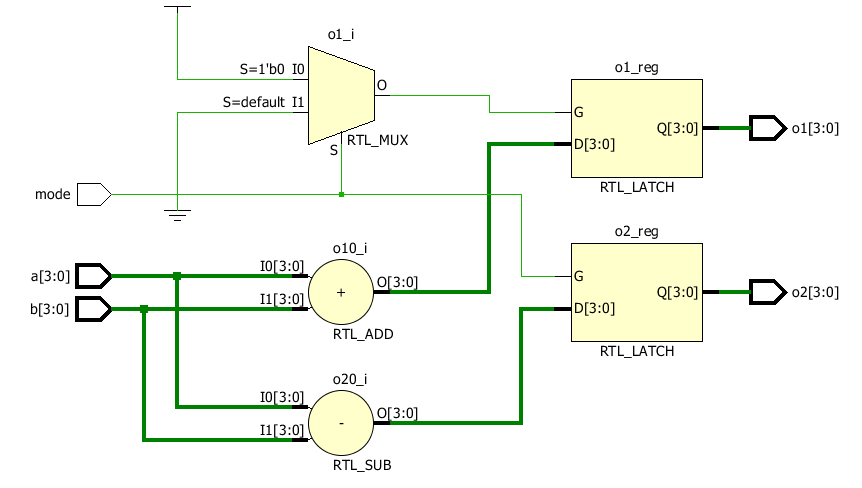
if (!mode)

o1 = a + b;

else

o2 = a - b;

endmodule



## Пример - always\_comb

module always\_comb\_example(

input [3:0] a,b,

input mode,

output reg [3:0] o1, o2

);

always\_comb

if (!mode)

o1 = a + b;

else

o2 = a - b;

endmodule

* няма нужда от sensitivity list
* warnings

WARNING: [Synth 8-87] always\_comb on 'o1\_reg' did not result in combinational logic [C:/usr/Vivado/2017/examples/examples.srcs/sources\_1/imports/new/always\_comb\_example.sv:11]  
WARNING: [Synth 8-87] always\_comb on 'o2\_reg' did not result in combinational logic [C:/usr/Vivado/2017/examples/examples.srcs/sources\_1/imports/new/always\_comb\_example.sv:13]

### always\_ff

Описва последователни схеми.

always\_ff @(posedge clock iff reset == 0 or posedge reset) begin

r1 <= reset ? 0 : r2 + 1;

...

end

— съдържа един и само един **event control**

— променливите от лявата страна на оператори за присвояване не трябва да бъдат променяни от други процеси

### Procedural timing controls

timing control - определя **кога** ще се изпълнят процедурните оператори.

SystemVerilog има два типа на timing control.

* **delay control -** времето между “срещането” на оператора и момента когато оператора наистина се изпълнява
* **event control** - забавя изпълнението на оператора докато настъпи определено събитие. Събитието може да е промяна на стойността на верига или променлива. Пример за често използван event control e нарастващият фронт на тактов сигнал.

#### **Delay control**

A procedural statement following the delay control shall be delayed in its execution with respect to the

procedural statement preceding the delay control by the specified delay.

Example 1: The following example delays the execution of the assignment by 10 time units:

#10 rega = regb;

Example 2: The next examples provide an expression following the number sign (#). Execution of the

assignment is delayed by the amount of simulation time specified by the value of the expression.

#d rega = regb; // d is defined as a parameter

#((d+e)/2) rega = regb; // delay is average of d and e

#### **Event control**

The execution of a procedural statement can be synchronized with a **value change on a net or variable** or the occurrence of a declared event.

The event can also be based on the **direction of the change**, that is, toward the value 1 (**posedge**) or toward the value 0 (**negedge**). The behavior of posedge and negedge events can be described as follows:

— A **negedge** shall be detected on the transition from **1** to x, z, or 0, and from x or z to 0

— A **posedge** shall be detected on the transition from **0** to x, z, or 1, and from x or z to 1

< Примери: counter.sv, timer\_switch.sv >