Моделиране на последователни схеми

[Регистри](#_hysiui88j09j)

[Тригер, управляван по фронт](#_vp7csloy2uky)

[n-битов регистър](#_44pape3plua8)

[Фиксатори](#_a7pw1e1kul1j)

[Преместващи регистри](#_mdfmueufko9d)

[Пример 1](#_chvu3f318ri3)

[Пример 2](#_6y59osvgdncz)

[Броячи](#_roxiegu7ynlk)

[Пример - 4 битов брояч](#_yernkxfcjyz7)

Последователни (sequential) схеми - съдържат тригери

Изходите зависят от предишното състояние на схемата. Тригерите запазват състояние.

Само промени на определени входове водят до промени в изходите. Тригерите са чувствително само към такт и reset.

* **always\_ff** декларира намерение да се използват тригери управлявани по фронт (flip-flops)
* **always\_ff** трябва да съдържа **clock и reset** в списъка на активиращите сигнали (sensitivity list)
* използват се **не-блокиращи** присвоявания

## Тригер, управляван по фронт

module flip\_flop(

 input d, rst\_n, clk,

 output logic q

 );

 always\_ff @(posedge clk, negedge rst\_n)

 if(~rst\_n)

 q **<=** ‘0;

 else

 q **<=** d;

endmodule

# Регистри

## n-битов регистър

module register #(parameter N)(

 input [N-1:0] din,

 input rst\_n, clk, load,

 output logic [N-1:0] dout

 );

 always\_ff @(posedge clk, negedge rst\_n)

 if(~rst\_n)

 dout <= ‘0;

 else

 dout <= load ? din : dout;

 // TODO - example with ‘1

endmodule

# Фиксатори

**module** latch\_example **(**

 **input** **[**3**:**0**]** d\_in**,**

 **input** enable**,**

 **output** **logic** **[**3**:**0**]** q\_out**);**

 **always\_latch**

 **if(**enable**)** q\_out **<=** d\_in**;**

**endmodule**





# Преместващи регистри

## Пример 1

module shift\_register #(parameter N = 8) (

 input clock,

 input [2:0] mode,

 input [N-1:0] data\_in,

 output logic [N-1:0] data\_out

);

always\_ff @(posedge clock)

 case(mode)

 3'b000: data\_out <= ‘0; // clear

 3'b111: data\_out <= data\_in; // load

 3'b100: data\_out <= data\_out << 1; // shift left

 3'b001: data\_out <= data\_out >> 1; // shift right

 default: data\_out <= data\_out; // hold

 endcase

endmodule

## Пример 2

module lab1 #(parameter W = 8) (

 input clock, reset,

 output logic [W-1:0] sr

);

always\_ff @(posedge clock, posedge reset)

 if(reset)

 sr <= ‘0;

 else

 sr <= {sr[W-2:0], ~sr[W-1]};

endmodule

# Броячи

## Пример - 4 битов брояч

module cnt4 (

 input clk, reset\_n,

 output logic [3:0] q\_out);

 always\_ff @(posedge clk, negedge reset\_n)

 if(~reset\_n)

q\_out <= '0;

 else

q\_out <= q\_out + 1;

endmodule



**Десетичен брояч**

| **non-blocking assignment** | **blocking assignment** |
| --- | --- |
| module counter\_dec #(parameter N = 4) ( input clock, input reset, output logic [N-1:0] cnt);always\_ff @(posedge clock, posedge reset) if (reset) cnt <= 0; else begin cnt <= cnt+1; if (cnt==9) cnt <= 0; endendmodule | module counter\_dec #(parameter N = 4) ( input clock, input reset, output logic [N-1:0] cnt);always\_ff @(posedge clock, posedge reset) if (reset) cnt = 0; else begin cnt = cnt+1; if (cnt==10) cnt = 0; endendmodule |

always\_ff @(posedge clock, posedge reset)

 if (reset)

 cnt <= 0;

 else if (cnt==9)

 cnt <= 0;

 else

 cnt <= cnt+1;

endmodule