Моделиране на комбинационни схеми

# Декодери

[Декодери](#_j1ozwx5r3b09)

[Моделиране на ниво логически елемент (gate-level modeling)](#_l1dqu6owjwhx)

[Пример - декодер 2x4](#_17dp8vu)

[Логически елементи](#_3rdcrjn)

[Dataflow моделиране](#_xfcybxb9utis)

[Пример - декодер 2x4](#_1fob9te)

[Поведенческо моделиране (Behavioral мodeling)](#_26in1rg)

[Пример - декодер 2x4](#_ewgjn4xp6qxn)

[Мултиплексори](#_zf75noekz3g)

[Пример - мултиплексор 2x1 с оператор if](#_k3alip4acwu4)

[Пример - мултиплексор 2x1 - dataflow стил](#_nx6stbgqbsx3)

[Пример - мултиплексор 4x1 с оператор case](#_1ksv4uv)

[Компаратор](#_p4520bwgsych)

[Генератор на бит за нечетност](#_7rwzqkfeq2rl)

[Приоритетен Енкодер](#_92qe0610dpua)

[Пример - if](#_jbjta8n2k4lh)

[Пример - for](#_add1f2uln2ii)

[Суматор](#_ju2e91z5ffs0)

## Моделиране на ниво логически елемент (gate-level modeling)

### Пример - декодер 2x4

| // Gate level style  module decoder\_2x4\_v1(  output [3:0] D,  input A, B, enable  );  logic A\_not, B\_not, E\_not;  not G1 (A\_not, A);  not G2 (B\_not, B);  not G3 (E\_not, enable);  nand G4 (D[3], A\_not, B\_not, E\_not);  nand G5 (D[2], A\_not, B, E\_not);  nand G6 (D[1], A, B\_not, E\_not);  nand G7 (D[0], A, B, E\_not);  endmodule |  |
| --- | --- |

### Логически елементи

and, nand, or, nor, xor, xnor

buf, not

bufif1, bufif0, notif1, notif0 - три състояния

## **Dataflow** моделиране

### Пример - декодер 2x4

| // Dataflow style  module decoder\_2x4\_v2 (  output [3:0] D,  input A, B, enable  );  assign D[3] = !(!A && !B && !enable);  assign D[2] = !(!A && B && !enable);  assign D[1] = !(A && !B && !enable);  assign D[0] = !(A && B && !enable);  endmodule | E A B - D  1 x x - 1111  0 0 0 - 0111  0 0 1 - 1011  0 1 0 - 1101  0 1 1 - 1110 |
| --- | --- |

## Поведенческо моделиране (Behavioral мodeling)

**always\_comb + procedural assignment statements**

### Пример - декодер 2x4 (bin -> “hot” zero)

// Behavioral style

module decoder\_2x4\_v3 (

output logic [3:0] D,

input A, B, enable

);

always\_comb

if (~enable)

D = 4'b1111;

else

case ({A,B})

2'b00: D = 4'b0111;

2'b01: D = 4'b1011;

2'b10: D = 4'b1101;

2'b11: D = 4'b1110;

endcase

endmodule

# Мултиплексори

## Пример - мултиплексор 2x1 с оператор if

| module mux\_2x1(  output logic Y,  input A, B, select  );  always\_comb  if (select) Y = A;  else Y = B;  endmodule |  |
| --- | --- |

## Пример - мултиплексор 2x1 - dataflow стил

module mux\_2x1\_df(

output Y,

input A, B, select

);

assign Y = select ? A : B;

endmodule

## Пример - мултиплексор 4x1 с оператор case

| module mux\_4x1(  output logic Y,  input A, B, C, D,  input [1:0] select  );  always\_comb  case (select)  2'b00: Y = A;  2'b01: Y = B;  2'b10: Y = C;  2'b11: Y = D;  endcase  endmodule |  |
| --- | --- |

# Компаратор

// Dataflow description of a four-bit comparator

module comparator #(parameter N=4) (

output A\_lt\_B, A\_eq\_B, A\_gt\_B,

input [N-1:0] A, B

);

assign A\_lt\_B = A < B;

assign A\_gt\_B = A > B;

assign A\_eq\_B = A == B;

endmodule

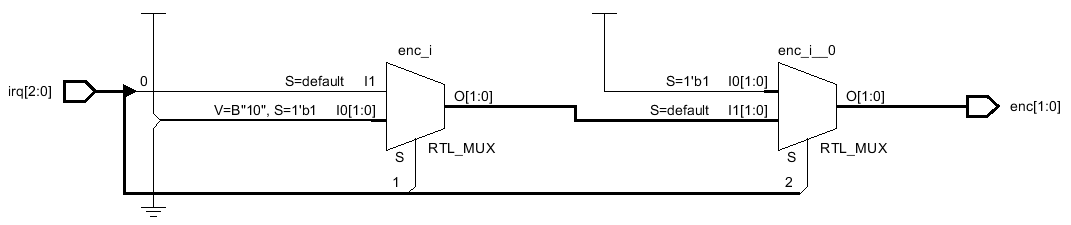
# Генератор на бит за нечетност

| module parity #(parameter SIZE = 4)(  input [SIZE-1:0] data,  output odd  );  assign odd = ^data;  endmodule | 0111 -> 1  0000 -> 0  0001 -> 1  0101 -> 0  0100 -> 1  0110 -> 0 |  |
| --- | --- | --- |

# Приоритетен Енкодер

## Пример - if

| module priority\_encoder(  input [2:0] irq,  output logic [1:0] enc  );  always\_comb  if (irq[2])  enc = 3;  else if (irq[1])  enc = 2;  else if (irq[0])  enc = 1;  else  enc = 0;  endmodule  Симулация  111 -> 3  100 -> 3  001 -> 1  010 -> 2  011 -> 2  000 -> 0 | module priority\_encoder\_test;  logic [2:0] irq;  logic [1:0] enc;  priority\_encoder **uut** (.\*);  initial $monitor("%b -> %d", irq, enc);  initial begin  irq = 3'b111;  #1 assert (enc==3);  irq = 3'b100;  #1 assert (enc==3);  irq = 3'b001;  #1 assert (enc==1);  irq = 3'b010;  #1 assert (enc==2);  irq = 3'b011;  #1 assert (enc==2);  irq = 3'b000;  #1 assert (enc==0);  $finish;  end  endmodule |
| --- | --- |



## Пример - for

module priority\_encoder(

input [2:0] irq,

output logic [1:0] enc

);

always\_comb begin

enc = 0;

for(int i=2; i>=0; i-- ) begin

if (irq[i]) begin

enc = i+1;

break;

end

end

end

endmodule

# Суматор

module adder #(parameter N = 16)(

output [N-1:0] s,

output cout,

input [N-1:0] a, b,

input cin

);

assign {cout, s} = a + b + cin;

endmodule

