2x4 decoder

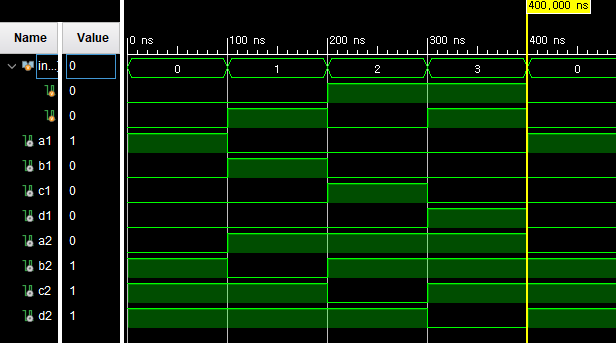
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | **Output** | | | |
| A | B | D0 | D1 | D2 | D3 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

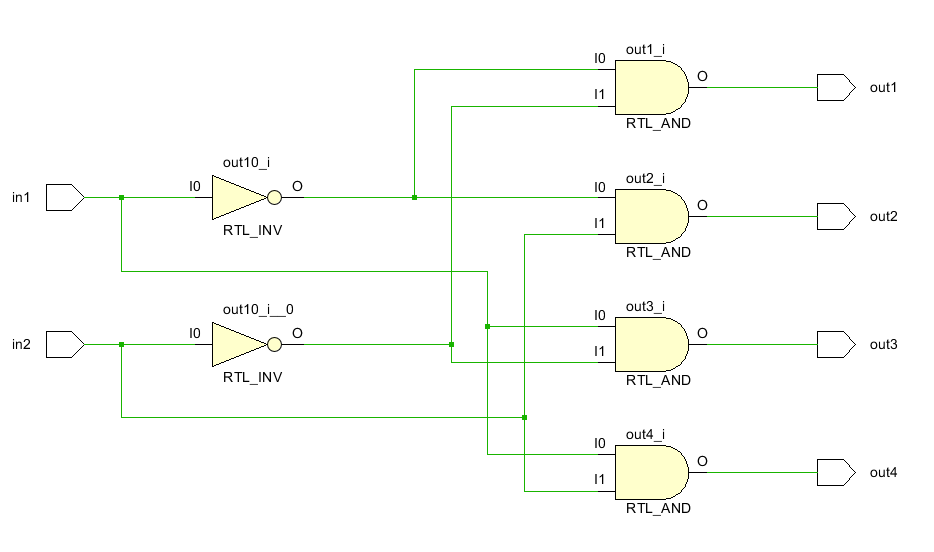
Active high

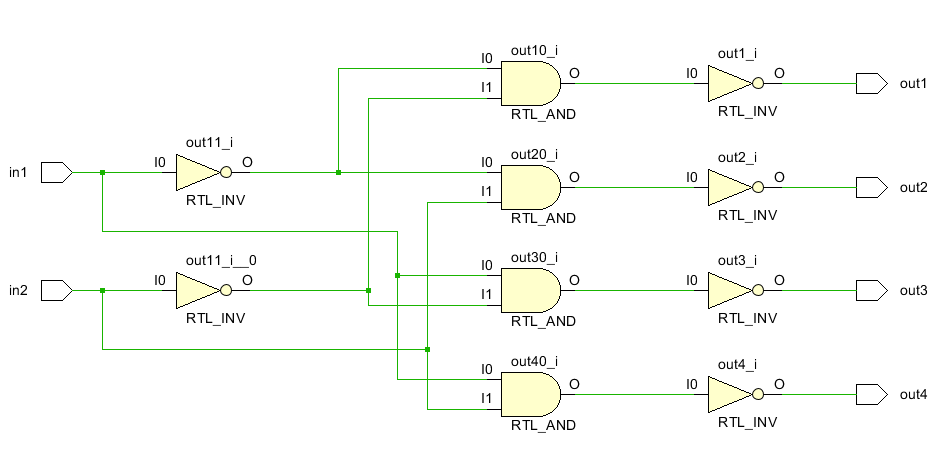
Active low

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | **Output** | | | |
| A | B | D0 | D1 | D2 | D3 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

Sim

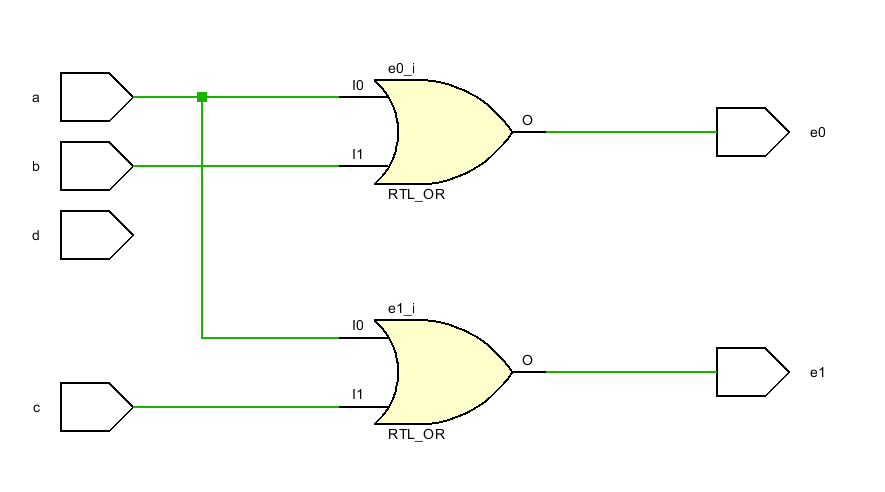


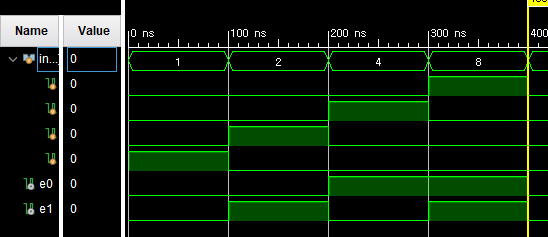




4x2 encoder

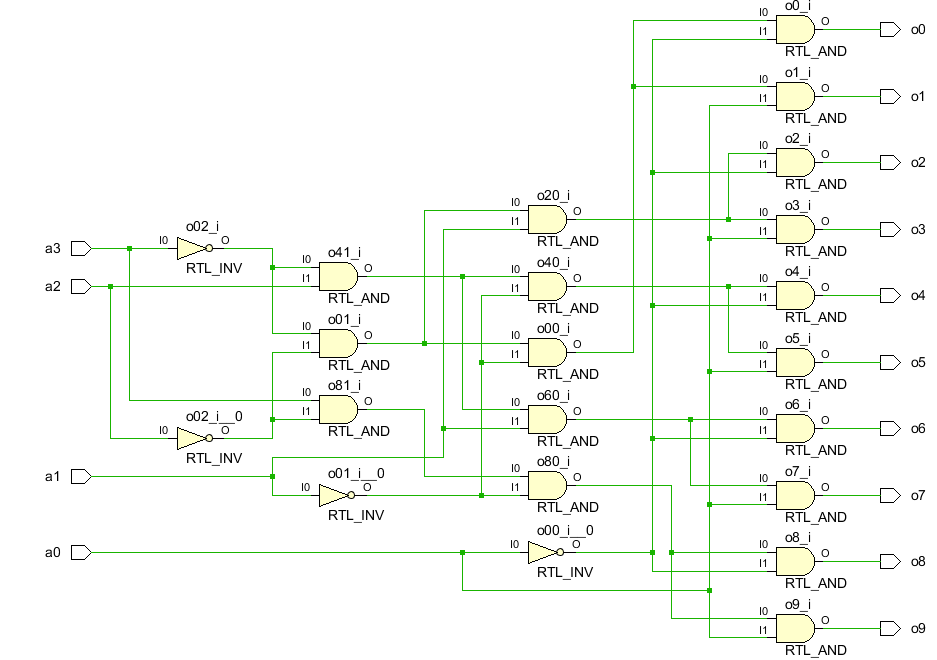
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | | | **Output** | |
| A | B | C | D | E0 | E1 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |

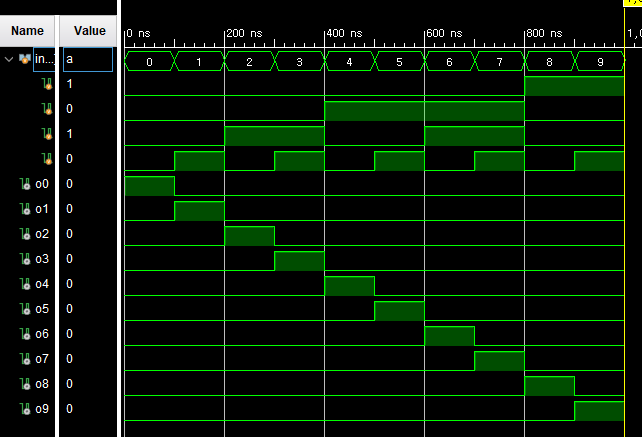




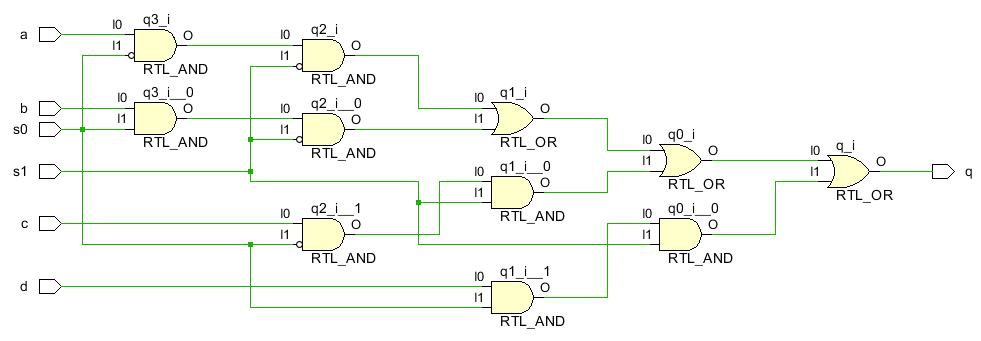
ｂｃｄ　ｔｏ　ｄｅｃｉｍａｌ

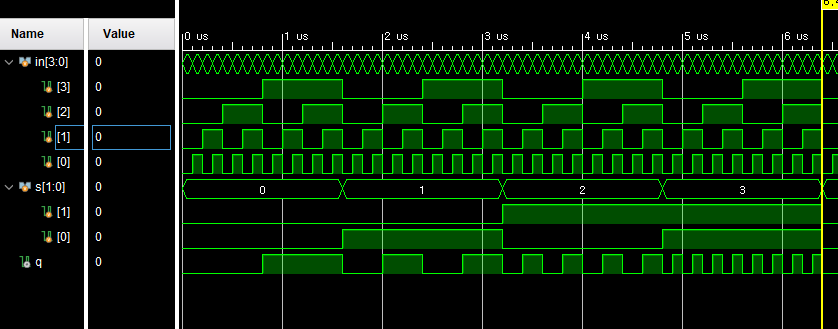
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a3 | a2 | a1 | a0 | e0 | e1 | e2 | e3 | e4 | e5 | e6 | e7 | e8 | e9 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |





４－１　ｍｕｘ

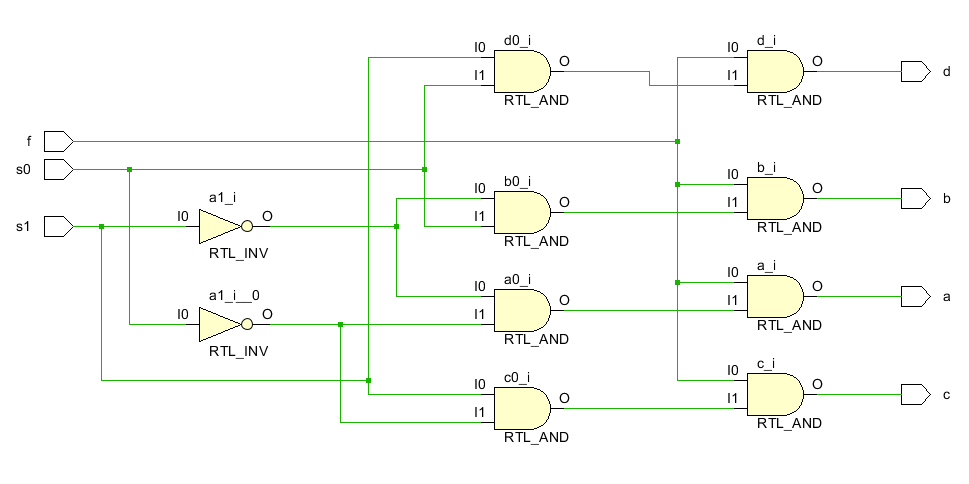


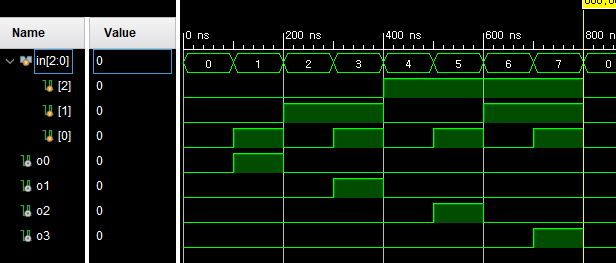


|  |  |  |
| --- | --- | --- |
| s1 | s0 | q |
| 0 | 0 | a |
| 0 | 1 | b |
| 1 | 0 | c |
| 1 | 1 | d |

１－４　ｄｅｍｕｘ

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Select** | | **Data** | **Output** | | | |
| ｓ１ | ｓ０ | Ｆ | Out０ | Out１ | Out２ | Out３ |
| ０ | ０ | ０ | ０ | ０ | ０ | ０ |
| ０ | ０ | １ | １ | ０ | ０ | ０ |
| ０ | １ | ０ | ０ | ０ | ０ | ０ |
| ０ | １ | １ | ０ | １ | ０ | ０ |
| １ | ０ | ０ | ０ | ０ | ０ | ０ |
| １ | ０ | １ | ０ | ０ | １ | ０ |
| １ | １ | ０ | ０ | ０ | ０ | ０ |
| １ | １ | １ | ０ | ０ | ０ | １ |





`timescale 1ns / 1ps

module ah\_decoder(

input in1, in2,

output out1, out2, out3, out4

);

assign out1 = ~in1&~in2;

assign out2 = ~in1&in2;

assign out3 = in1&~in2;

assign out4 = in1&in2;

endmodule

module al\_decoder(

input in1, in2,

output out1, out2, out3, out4

);

assign out1 = ~(~in1&~in2);

assign out2 = ~(~in1&in2);

assign out3 = ~(in1&~in2);

assign out4 = ~(in1&in2);

endmodule

module encoder (

input a, b, c, d,

output e0, e1

);

assign e0 = a|b;

assign e1 = a|c;

endmodule

module bcd2dec(

input a3, a2, a1, a0,

output o0, o1, o2, o3, o4, o5, o6, o7, o8, o9

);

assign o0 = ~a3&~a2&~a1&~a0;

assign o1 = ~a3&~a2&~a1&a0;

assign o2 = ~a3&~a2&a1&~a0;

assign o3 = ~a3&~a2&a1&a0;

assign o4 = ~a3&a2&~a1&~a0;

assign o5 = ~a3&a2&~a1&a0;

assign o6 = ~a3&a2&a1&~a0;

assign o7 = ~a3&a2&a1&a0;

assign o8 = a3&~a2&~a1&~a0;

assign o9 = a3&~a2&~a1&a0;

endmodule

module mux41(

input a, b, c, d, s1, s0,

output q

);

assign q = (a&~s0&~s1)|(b&s0&~s1)|(c&~s0&s1)|(d&s0&s1);

endmodule

module demux14(

input s1, s0, f,

output a, b, c, d

);

assign a = f&(~s1&~s0);

assign b = f&(~s1&s0);

assign c = f&(s1&~s0);

assign d = f&(s1&s0);

endmodule

ｓｉｍ

`timescale 1ns / 1ps

module decodersim;

reg [1:0] in;

wire a1, b1, c1, d1, a2, b2, c2, d2;

ah\_decoder ah(in[1], in[0], a1, b1, c1, d1);

al\_decoder al(in[1], in[0], a2, b2, c2, d2);

initial in = 2'b00;

always in = #100 in+1;

initial begin

#1600

$finish;

end

endmodule

module encodersim;

reg [3:0] in;

wire e0, e1;

encoder en(in[3], in[2], in[1], in[0], e0, e1);

initial in = 4'b0001;

always in = #100 in\*2;

initial begin

#1600

$finish;

end

endmodule

module b2dsim;

reg [3:0] in;

wire o0, o1, o2, o3, o4, o5, o6, o7, o8, o9;

bcd2dec b2d(in[3], in[2], in[1], in[0], o0, o1, o2, o3, o4, o5, o6, o7, o8, o9);

initial in = 4'b0000;

always in = #100 in+1;

initial begin

#1600

$finish;

end

endmodule

module muxsim;

reg [3:0] in;

reg [1:0] s;

wire q;

mux41 mux(in[3], in[2], in[1], in[0], s[1], s[0], q);

initial in = 4'b0000;

initial s = 2'b00;

always in = #100 in+1;

always s = #1600 s+1;

initial begin

#1600

$finish;

end

endmodule

module demuxsim;

reg [2:0] in;

wire o0, o1, o2, o3;

demux14 dmux(in[2], in[1], in[0], o0, o1, o2, o3);

initial in = 3'b000;

always in = #100 in+1;

initial begin

#1600

$finish;

end

endmodule