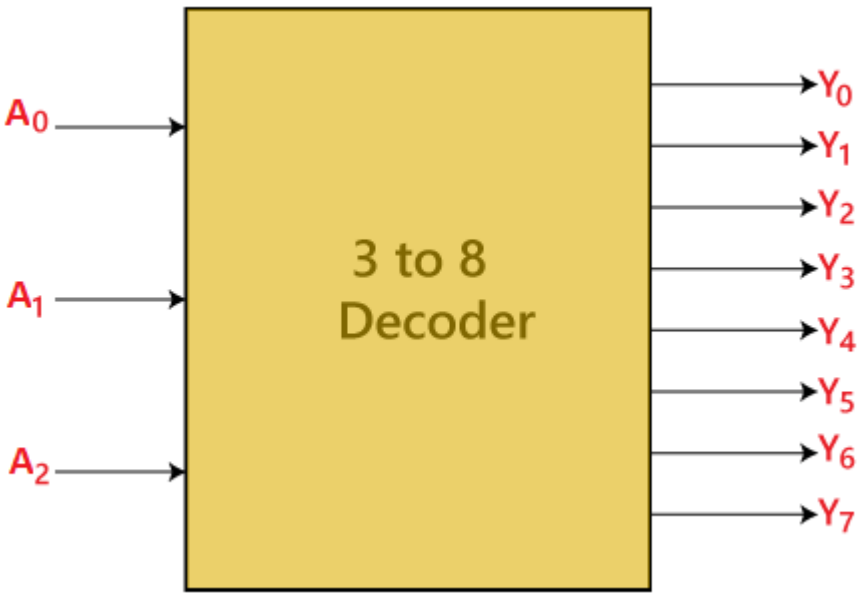
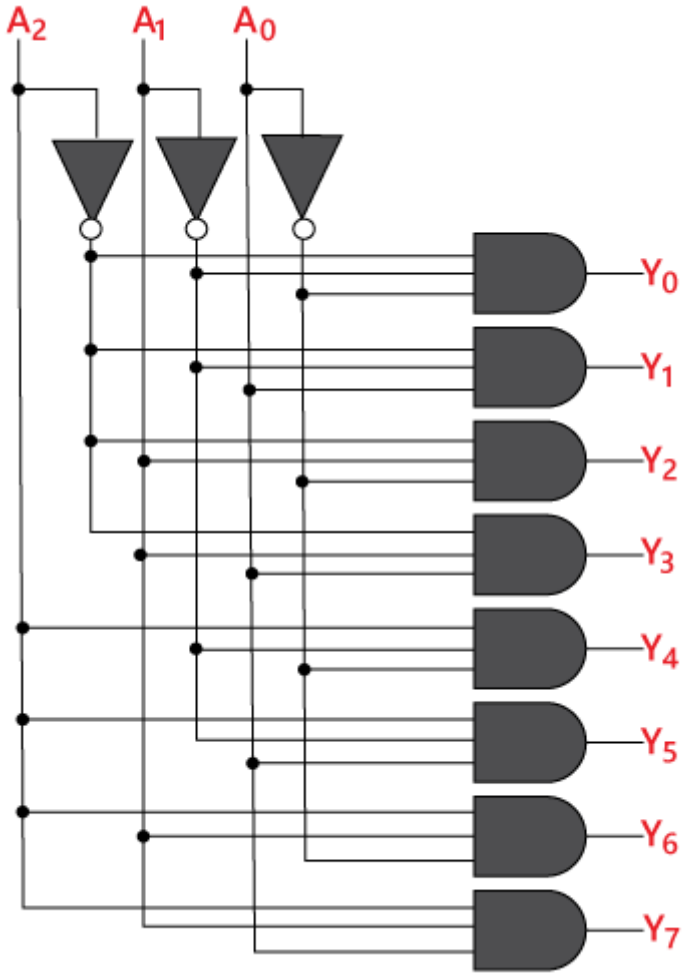
آزمایشگاه سیستم های دیجیتال2

آزمایش 1

نمایشگر هفت قسمتی

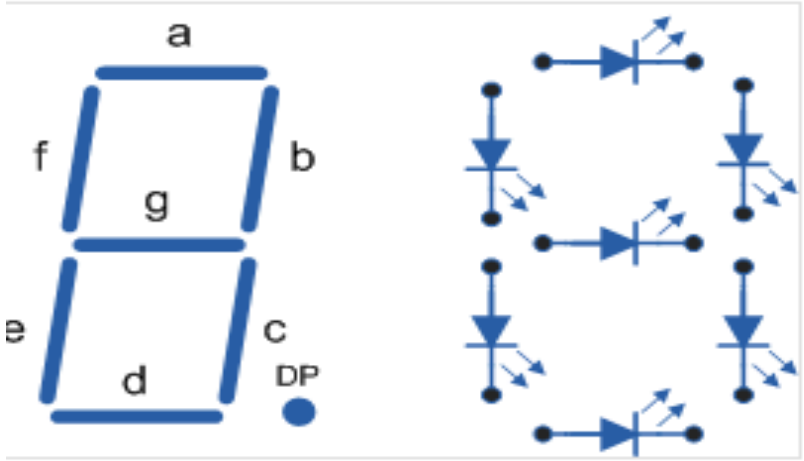
Decoder

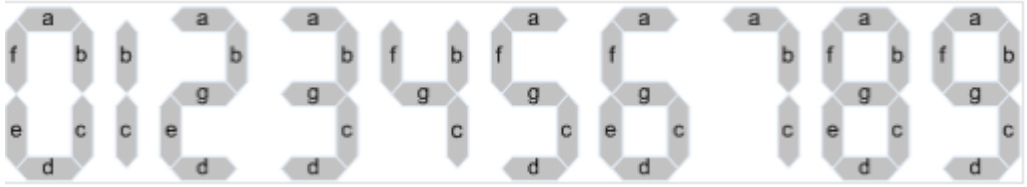

A0
A1
A2
3 to 8
Decoder
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7


A2
A1
A0
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7

Seven segment

- Structure





کاراکتر متناظر	سگمنت‌های نمایشگر						
	a	b	c	d	e	f	g
0	×	×	×	×	×	×	
1		×	×				
2	×	×		×	×		×
3	×	×	×	×			×
4		×	×			×	×
5	×		×	×		×	×
6	×		×	×	×	×	×
7	×	×	×				

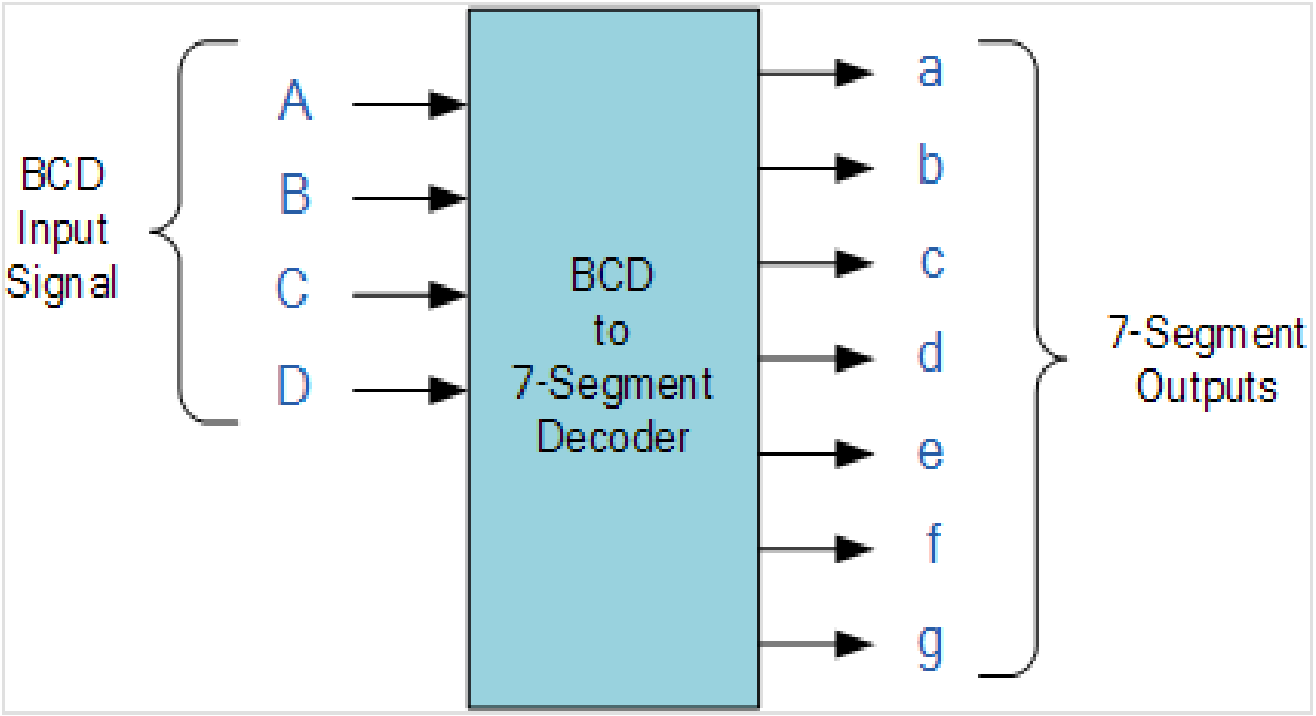
کاراکتر متناظر	سگمنت‌های نمایشگر						
	a	b	c	d	e	f	g
8	×	×	×	×	×	×	×
9	×	×	×	×		×	×
A	×	×	×		×	×	×
b			×	×	×	×	×
C	×			×	×	×	
d		×	×	×	×		×
E	×			×	×	×	×
F	×				×	×	×

Binary Coded Decimal(BCD)

دهدهی	الگوی باینری				BCD
	8	4	2	1	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7

دهدهی	الگوی باینری				BCD
	8	4	2	1	
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	Invalid
11	1	0	1	1	Invalid
12	1	1	0	0	Invalid
13	1	1	0	1	Invalid
14	1	1	1	0	Invalid
15	1	1	1	1	Invalid

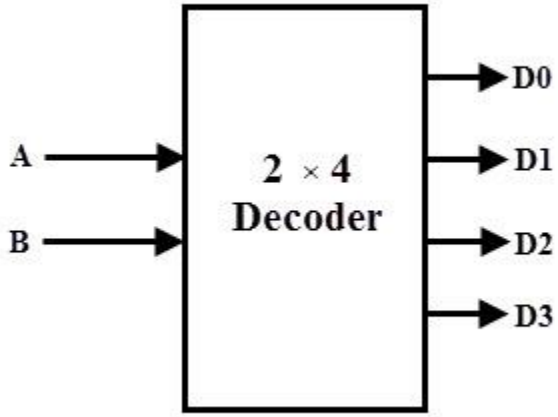
BCD decoding to seven segment



Assign Statements

Nested Condition Operations

Describing a Decoder



Assign Statements

Nested Condition Operations

Describing a Decoder

```
`timescale 1ns/100ps

module dcd2_4 (input a, b, output d0, d1, d2, d3 );
   assign
      {d3, d2, d1, d0} = ( {a, b} == 2'b00 ) ? 4'b0001 :
                         ( {a, b} == 2'b01 ) ? 4'b0010 :
                         ( {a, b} == 2'b10 ) ? 4'b0100 :
                         ( {a, b} == 2'b11 ) ? 4'b1000 :
                         4'b0000;
endmodule
```

Assign Statements

Nested Condition Operations

Test for Decoder

```
module test_dcd;

    // Inputs
    reg a;
    reg b;
    // Outputs
    wire d0;
    wire d1;
    wire d2;
    wire d3;
    // Instantiate the Unit Under Test
    dcd2_4 uut (
        .a(a),
        .b(b),
        .d0(d0),
        .d1(d1),
        .d2(d2),
        .d3(d3)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        #50 a = 0; b = 1;
        #50 a = 1; b = 0;
        #50 a = 1; b = 1;
        // Wait 100 ns for global reset
        #100;
        // Add stimulus here
    end
endmodule
```

Assign Statements

Procedural case statement

Decoder Using case Statement

```
module dcd2_4 (input a, b, output reg d0, d1, d2, d3 );

    always @(a, b) begin
        case ( { a, b } )
            2'b00 : { d3, d2, d1, d0 } = 4'b0001;
            2'b01 : { d3, d2, d1, d0 } = 4'b0010;
            2'b10 : { d3, d2, d1, d0 } = 4'b0100;
            2'b11 : { d3, d2, d1, d0 } = 4'b1000;
            default: { d3, d2, d1, d0 } = 4'b0000;
        endcase
    end

endmodule
```

Assign Statements

Procedural case statement

Decoder Using case Statement

```
module test_dcd;

    // Inputs
    reg a;
    reg b;
    // Outputs
    wire d0;
    wire d1;
    wire d2;
    wire d3;
    // Instantiate the Unit Under Test
    dcd2_4 uut (
        .a(a),
        .b(b),
        .d0(d0),
        .d1(d1),
        .d2(d2),
        .d3(d3)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;
        #50 a = 0; b = 1;
        #50 a = 1; b = 0;
        #50 a = 1; b = 1;
        // Wait 100 ns for global reset
        #100;
        // Add stimulus here
    end
endmodule
```