1. **4:1 Mux**

**Code:**

`timescale 1ns / 1ps

module mux4x1(

input [3:0]data,

input [1:0]sel,

output reg out

);

always@(\*)

begin

case (sel)

2'b00: out = data[0];

2'b01: out = data[1];

2'b10: out = data[2];

2'b11: out = data[3];

endcase

end

endmodule

**Testbench:**

`timescale 1ns / 1ps

module t\_mux4x1;

// Inputs

reg [3:0] data;

reg [1:0] sel;

// Outputs

wire out;

// Instantiate the Unit Under Test (UUT)

mux4x1 uut (

.data(data),

.sel(sel),

.out(out)

);

initial begin

// Initialize Inputs

data = 0000;

sel = 00;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

data = 4'b1010;

sel = 2'b00;

#50;

sel = 2'b01;

#50;

sel = 2'b10;

#50;

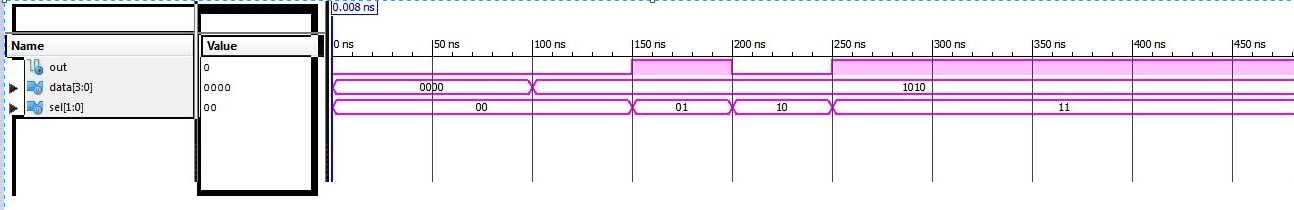
sel = 2'b11;

#50;

end

endmodule

**Waveform:**



1. **Full Adder**

**Code:**

`timescale 1ns / 1ps

module fulladder(

input wire a,

input wire b,

input wire cin,

output wire sum,

output wire cout,

wire p,

wire q,

wire r

);

xor x(sum,a,b,cin);

and m(p,a,b);

and n(q,cin,b);

and o(r,a,cin);

or l(cout,p,q,r);

endmodule

**Testbench:**

`timescale 1ns / 1ps

module t\_fulladder;

// Inputs

reg a;

reg b;

reg cin;

// Outputs

wire sum;

wire cout;

// Instantiate the Unit Under Test (UUT)

fulladder uut (

.a(a),

.b(b),

.cin(cin),

.sum(sum),

.cout(cout)

);

initial begin

// Initialize Inputs

a = 0;

b = 0;

cin = 0;

// Wait 100 ns for global reset to finish

#100;

// Add stimulus here

a = 1'b0;

b = 1'b1;

cin = 1'b0;

#50;

a = 1'b1;

b = 1'b1;

cin = 1'b0;

#50;

a = 1'b0;

b = 1'b0;

cin = 1'b1;

#50;

a = 1'b0;

b = 1'b0;

cin = 1'b0;

#50;

a = 1'b1;

b = 1'b1;

cin = 1'b1;

end

endmodule

**Waveform:**

