**EXPERIMENT 1(A)**

**Aim:** Design, Implement and Verify Basic Gates in Xilinx Vivado.

**Verilog Code**:

`timescale 1ns / 1ps

module Basic\_Gates(

    input a\_016, b\_016,

    output c1\_016, c2\_016, c3\_016, c4\_016, c5\_016, c6\_016, c7\_016

    );

    and(c1\_016, a\_016, b\_016);

    or(c2\_016, a\_016, b\_016);

    xor(c3\_016, a\_016, b\_016);

    xnor(c4\_016, a\_016, b\_016);

    nor(c5\_016, a\_016, b\_016);

    nand(c6\_016, a\_016, b\_016);

    not(c7\_016, a\_016);

endmodule

**EXPERIMENT 1(B)**

**Aim:**

**Verilog Code:**

//gate

module HA(

    input a\_016, b\_016,

    output s\_gate, c\_gate

    );

    xor(s\_gate, a\_016 b\_016);

    and(c\_gate, a\_016, b\_016);

endmodule

//data

module HA(

    input a\_016, b\_016,

    output s\_data, c\_data

    );

    assign s\_data = a\_016^b\_016;

    assign c\_data  = a\_016&b\_016;

endmodule

//conditional

module HA(

    input a\_016, b,

    output s\_cond, c\_cond

    )

    assign s\_cond = a\_016 ? (b\_016 ? 0 : 1) : (b\_016 ? 1 : 0);

    assign c\_cond = a\_016 ? (b\_016 ? 1 : 0) : 0;

endmodule

**EXPERIMENT 2**

**Aim:**

**Verilog code:**

// Exp 2

//Gate level

module FA\_1Bit(

    input a\_016, b\_016, cin\_016,

    output s\_016, cout\_016,

    wire y1\_016, y2\_016, y3\_016

);

    xor(y1\_016, a\_016, b\_016);

    and(y2\_016, y1\_016, cin\_016);

    and(y3\_016, a\_016, b\_016);

    xor(s\_016, y1\_016, cin\_016);

    or(cout\_016, y2\_016, y3\_016);

endmodule

//Data level

module FA\_1Bit(

    input a\_016, b\_016, cin\_016,

    output s\_016, cout\_016,

    wire y1\_016, y2\_016, y3\_016

);

    assign y1\_016 = a\_016^b\_016;

    assign y2\_016 = y1\_016&cin\_016;

    assign y3\_016 = a\_016&b\_016;

    assign s\_016 = y1\_016^cin\_016;

    assign cout\_016 = y2\_016|y3\_016;

endmodule

//Conditional level

module FA\_1Bit(

    input a\_016, b\_016, cin\_016,

    output s\_016, cout\_016,

    wire y1\_016, y2\_016, y3\_016

);

    assign y1\_016 = a\_016 ? 1 : (b\_016 ? : 1 : 0);

    assign y2\_016 = y1\_016 ? (cin\_016 ? : 1 : 0) : 0;

    assign y3\_016 = a\_016 ? (b\_016 ? : 1 : 0) : 0;

    assign s\_016 = y1\_016 ? 1 : (cin\_016 ? : 1 : 0);

    assign cout\_016 = y2\_016 ? 1 : (y3\_016 ? : 1 : 0);

endmodule

**EXPERIMENT 3**

**Aim:**

**Verilog code:**

module FA\_HA(

    input a\_016, b\_016, cin\_016,

    output s\_016, cout\_016,

    wire y1\_016, y2\_016, y3\_016

);

    HA i1(a\_016, b\_016, y1\_016, y2\_016);

    HA i2(y1\_016, cin\_016, s\_016, y3\_016);

    or(cout\_016, y2\_016, y3\_016);

endmodule

module HA(

    input a\_016, b\_016,

    output s\_016, c\_016

);

    xor(s\_016, a\_016, b\_016);

    and(c\_016, a\_016, b\_016);

endmodule

--------------------------------------------

// Exp 3 b

module MUX(

    input I0, I1, I2, I3, s1, s0,

    output y\_016

);

    assign y\_016 = s1\_016 ? (s0\_016 ? I3 : I2) : (s0\_016 ? I1 : I0);

endmodule

**EXPERIMENT 4**

**Aim:**

**Verilog code:**

module MUX\_16x1(

input [15:0] I,

input [3:0] S,

output Y

);

wire [3:0]w;

MUX4x1 M1(.I(I[3:0]), .S(S[1:0]),.Y(w[0]));

MUX4x1 M2(.I(I[7:4]), .S(S[1:0]),.Y(w[1]));

MUX4x1 M3(.I(I[11:8]), .S(S[1:0]),.Y(w[2]));

MUX4x1 M4(.I(I[15:12]), .S(S[1:0]),.Y(w[3]));

MUX4x1 M5(.I(w), .S(S[3:2]),.Y(Y));

endmodule

module MUX4x1(

input [3:0] I,

input [1:0] S,

output Y

);

wire [1:0]w;

MUX2x1 M1(.I(I[1:0]), .S(S[0]),.Y(w[0]));

MUX2x1 M2(.I(I[3:2]), .S(S[0]),.Y(w[1]));

MUX2x1 M3(.I(w), .S(S[1]),.Y(Y));

endmodule

module MUX2x1(

input [1:0] I,

input S,

output Y

);

assign Y = S ? I[1] : I[0];

endmodule