Maven Lab 2

1. Operators

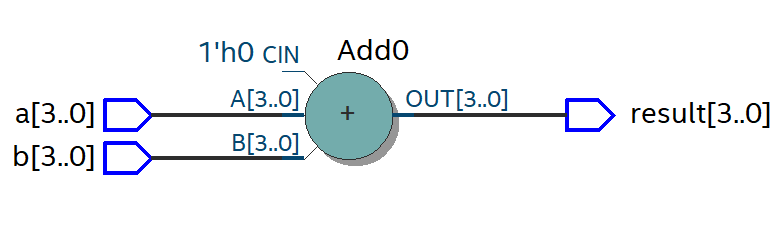
CODE:SUM

module sum(input [3:0]a,input [3:0]b,output [3:0]result);

assign result=a+b;

endmodule

NETLIST:



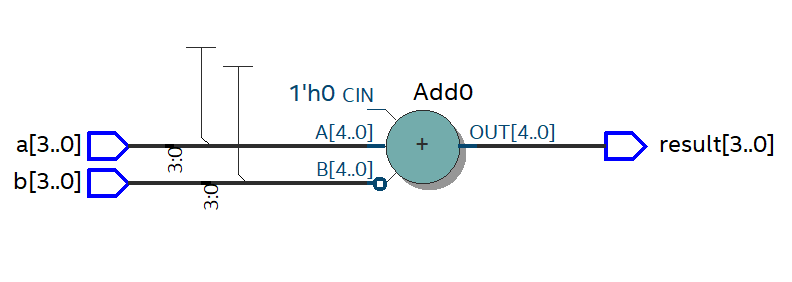
CODE: SUBTRACTOR

module sub(input [3:0]a,input [3:0]b,output [3:0]result);

assign result=a-b;

endmodule

NETLIST:



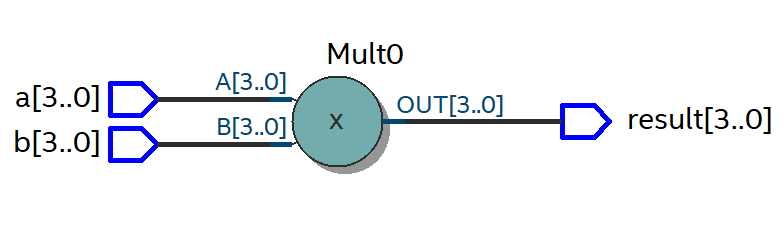
CODE: MULTIPLICATION

module mul(input [3:0]a,input [3:0]b,output [3:0]result);

assign result=a\*b;

endmodule

NETLIST:



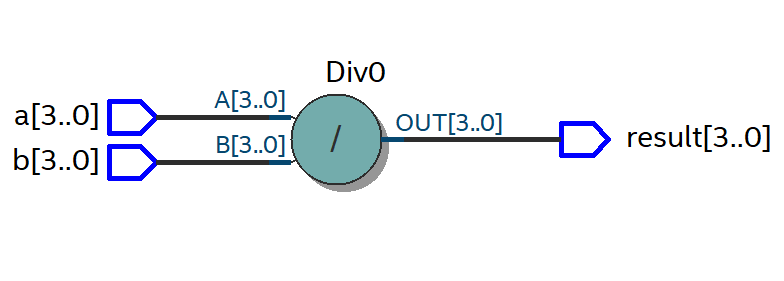
CODE: DIVISION

module div(input [3:0]a,input [3:0]b,output [3:0]result);

assign result=a/b;

endmodule

NETLIST:



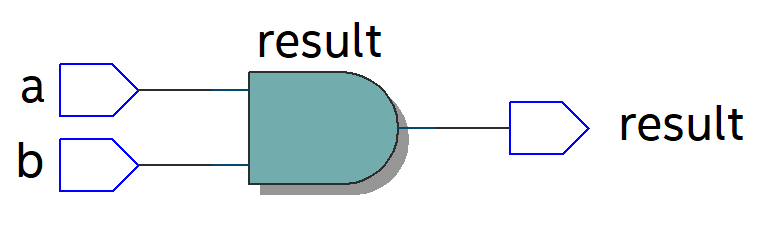
CODE: AND

module and\_gate(input a,input b,output result);

assign result=a&b;

endmodule

NETLIST:



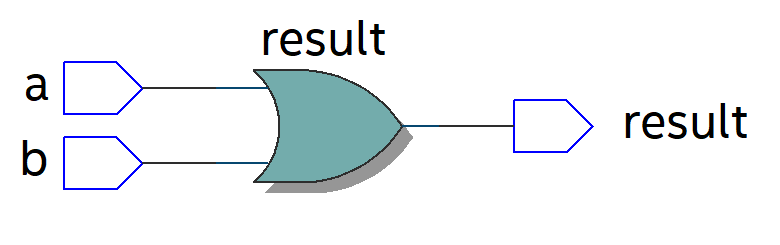
CODE: OR

module or\_gate(input a,input b,output result);

assign result=a|b;

endmodule

NETLIST:



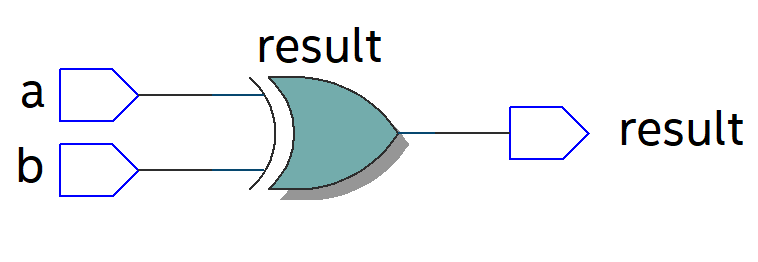
CODE: XOR

module xor\_gate(input a,input b,output result);

assign result=a^b;

endmodule

NETLIST:



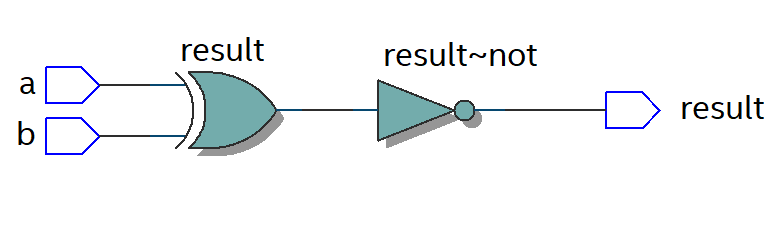
CODE: EQUAL

module equal\_op(input a,input b,output result);

assign result=(a==b);

endmodule

NETLIST:



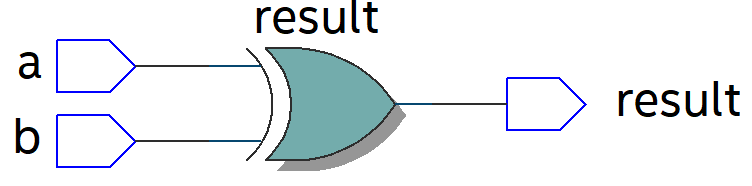
CODE: EQUAL

module equal\_op(input a,input b,output result);

assign result=(a==b);

endmodule

NETLIST:



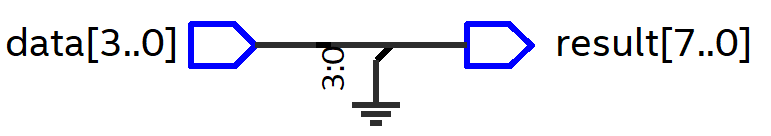
CODE:LEFT SHIFT

module Left\_Shift ( input [3:0] data, output [7:0] result);

assign result = data << 2;

endmodule

NETLIST:



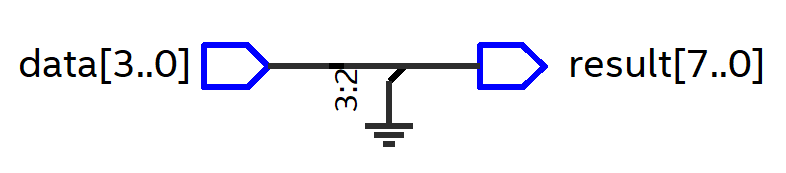
CODE:RIGHT SHIFT

module Right\_Shift ( input [3:0] data, output [7:0] result);

assign result = data >> 2;

endmodule

NETLIST:

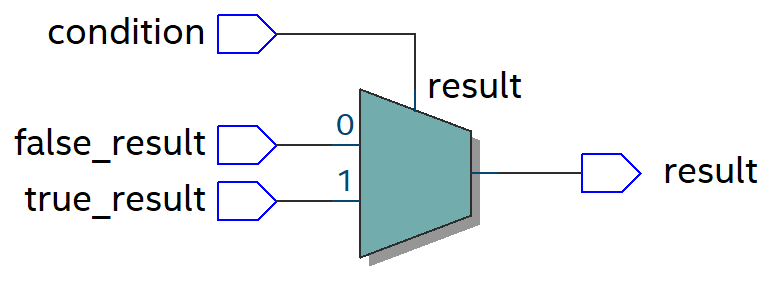


CODE:CONDITION OPERATOR

module Conditional\_Op( input condition,input true\_result,input false\_result,output result);

assign result = (condition) ? true\_result : false\_result;

endmodule



1. ALU

CODE:

module ALU(input [7:0]a,input [7:0]b,input [2:0]opcode,input clk,input rst,output reg [7:0]result);

always@(posedge clk)

begin

if(rst)

begin

result=8'b0;

end

else

begin

case(opcode)

3'b000: result<=a+b;

3'b001: result<=a-b;

3'b010: result<=a\*b;

3'b011: result<=a/b;

3'b100: result<=a&b;

3'b101: result<=a|b;

3'b110: result<=a^b;

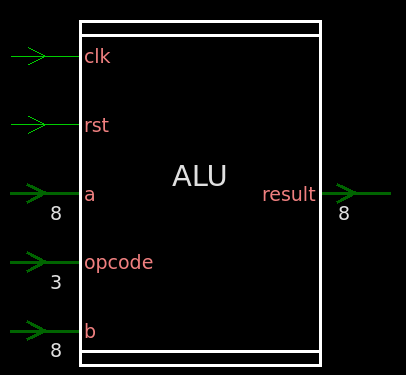
3'b111: result<=(a>b)?8'b1:8'b0;

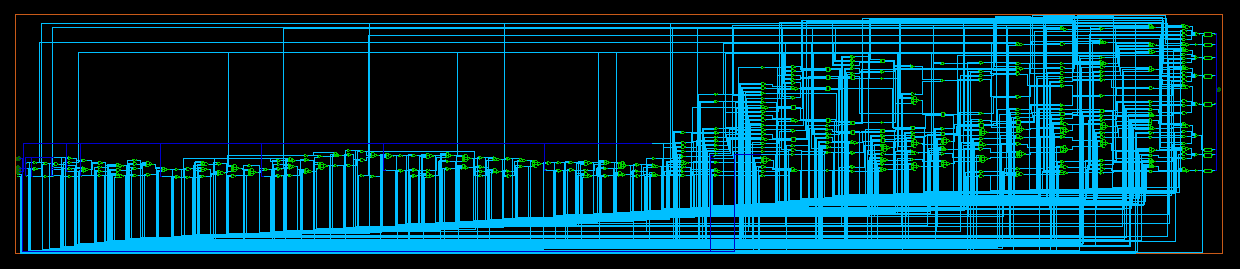
endcase

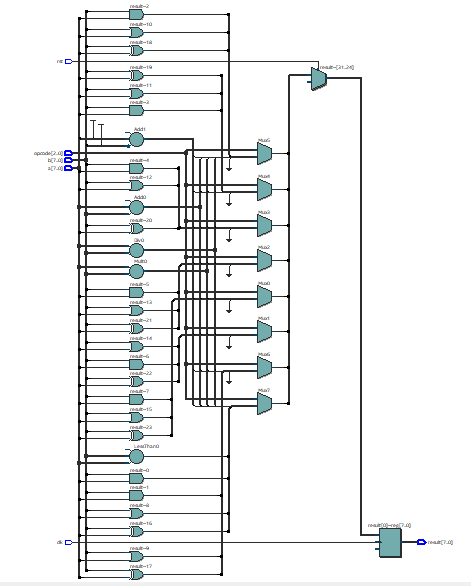
end

end

endmodule







TESTBENCH:

module ALU\_tb;

reg [7:0]a;

reg [7:0]b;

reg [2:0]opcode;

reg clk,rst;

wire [7:0]result;

integer i;

ALU dut(a,b,opcode,clk,rst,result);

initial

begin

a=8'b01100011;

b=8'b00011001;

clk=0;

rst=0;

opcode=8'b0;

end

always

begin

#5 clk=~clk;

end

initial

begin

rst=1;

#10 rst=0;

for(i=0;i<8;i=i+1)

begin

opcode = i;

#10;

end

end

endmodule

