1.

module FourBitMagComp(input[3:0] a, input[3:0] b, output eq, output lt, output gt);

assign eq = a == b;

assign lt = a < b;

assign gt = a > b;

endmodule

module EightBitMagComp(input[7:0] a, input[7:0] b, output eq, output lt, output gt);

wire tempEQ1, tempLT1, tempGT1;

wire tempEQ2, tempLT2, tempGT2;

FourBitMagComp first4BitMagComp(a[7:4], b[7:4], tempEQ1, tempLT1, tempGT1);

FourBitMagComp second4BitMagComp(a[3:0], b[3:0], tempEQ2, tempLT2, tempGT2);

assign eq = tempEQ1 & tempEQ2;

assign gt = tempGT1 | (tempEQ1 & tempGT2);

assign lt = ~eq & ~gt;

endmodule

module EightBitMagComp\_tb;

reg[7:0] a, b;

wire eq, lt, gt;

EightBitMagComp eightBitMagComp(a, b, eq, lt, gt);

initial begin

$monitor("%d a=%d, b=%d -> eq=%b, lt=%b, gt=%b", $time, a, b, eq, lt, gt);

#10 a=8'd10;

b=8'd9;

#10 a=8'd66;

b=8'd107;

#10 a=8'd49;

b=8'd49;

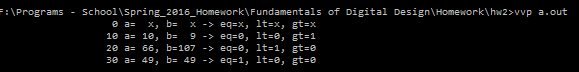
#10 $finish;

end

endmodule

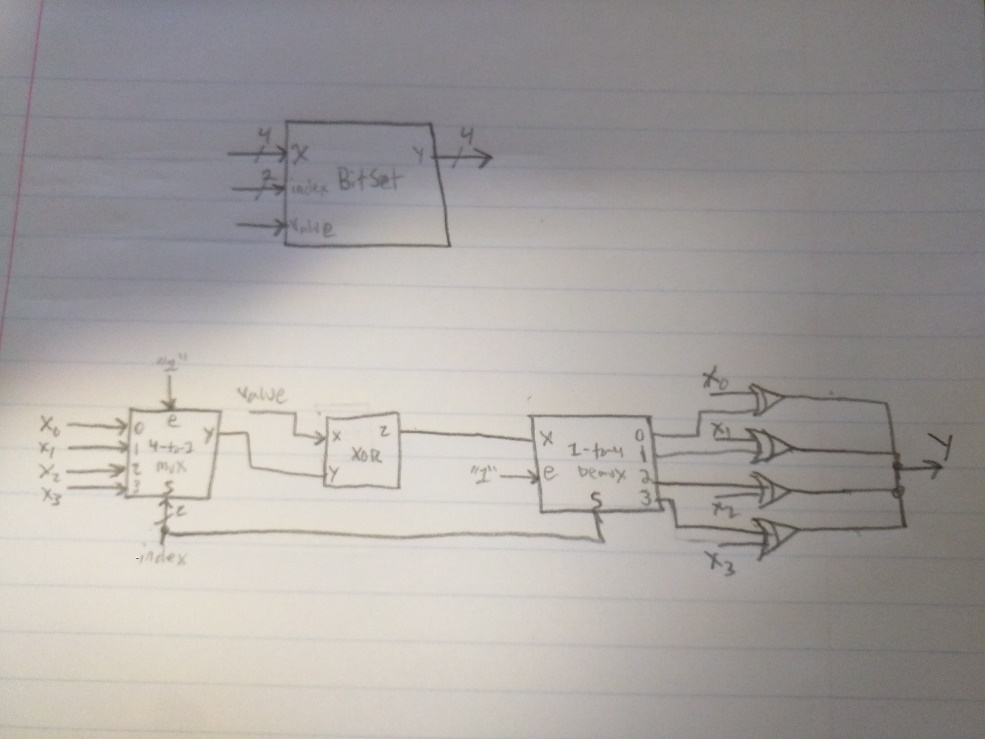
The design uses two 4-bit magnitude comparators in which the first compares the top 4 bits and the second compares the lower 4 bits. Equality is the simplest output because the two inputs are only equal if both 4-bit comparators return 1 in their EQ output. Greater than has two possibilities for determining if it is true or not. If the first 4-bit comparator returns a positive GT then “a” is guaranteed to be larger. The other possibility is if the high bits are equal and the second comparator returns GT then “a” is also larger. LT is determine by and-ing the inverses of EQ and LT because one of the three has to be true.

2.



3.

a)



b)

module BitSet(input[3:0] x, input[1:0] index, input value, output[3:0] y);

wire[3:0] demuxResult;

wire muxResult, temp;

Mux4To1 mux(1'b1, x, index, muxResult);

xor(temp, muxResult, value);

Demux1To4 demux(1'b1, temp, index, demuxResult);

xor(y[0], x[0], demuxResult[0]);

xor(y[1], x[1], demuxResult[1]);

xor(y[2], x[2], demuxResult[2]);

xor(y[3], x[3], demuxResult[3]);

endmodule

module Demux1To4(input e, input x, input[1:0] s, output[3:0] y);

reg y;

always @(e, x, s)

if(e)

case(s)

0: y = {3'b000, x};

1: y = {2'b00, x, 1'b0};

2: y = {1'b0, x, 2'b00};

3: y = {x, 3'b000};

endcase

else

y = 0;

endmodule

module Mux4To1(input e, input[3:0] x, input[1:0] s, output y);

assign y = x[s] & e;

endmodule

c)

module BitSet\_tb;

reg[3:0] x;

reg[1:0] index;

reg value;

wire[3:0] y;

BitSet bitSet(x, index, value, y);

initial begin

$monitor("%d x=%b%b%b%b index=%b%b value=%b --> y=%b%b%b%b", $time, x[3], x[2], x[1], x[0],

index[1], index[0], value, y[3], y[2], y[1], y[0]);

#10

x=4'b0000;

index=2'b00;

value=1'b1;

#10

x=4'b1111;

index=2'b01;

value=1'b0;

#10

x=4'b1010;

index=2'b10;

value=1'b0;

#10

x=4'b1110;

index=2'b11;

value=1'b1;

#10 $finish;

end

endmodule

d)

