module encoder(

input clk,

input reset,

input[31:0]msg,

output [55:0]O);

reg [55:0]O;

reg [3:0]D0 ;

reg [3:0]D1;

reg [3:0]D2;

reg [3:0]D3;

reg [3:0]D4;

reg [3:0]D5;

reg [3:0]D6;

reg [3:0]D7;

reg [1:3]R1;

reg [1:3]R2;

reg [1:3]R3;

reg [1:3]R4;

reg [1:3]R5;

reg [1:3]R6;

reg [1:3]R7;

reg [1:3]R8;

reg [0:7]m0;

reg [0:7]m1;

reg [0:7]m2;

reg [0:7]m3;

integer i;

always @(posedge clk)

begin

if(reset==1'b1)

begin

m0[0:7] = msg[31:24];

m1[0:7] = msg[23:16];

m2[0:7] = msg[15:8];

m3[0:7] = msg[7:0];

D7[3:0]={m3[7],m3[5],m2[6],m1[7]};

D6[3:0]={m3[6],m2[7],m1[0],m0[1]};

D5[3:0]={m0[0],m0[2],m1[1],m2[0]};

D4[3:0]={m3[4],m2[5],m1[6],m0[7]};

D3[3:0]={m3[3],m2[4],m1[5],m0[6]};

D2[3:0]={m3[2],m2[3],m1[4],m0[5]};

D1[3:0]={m3[1],m2[2],m1[3],m0[4]};

D0[3:0]={m3[0],m2[1],m1[2],m0[3]};

R1[1] = m1[7]^m2[6]^m3[7];

R1[2] = m1[7]^m3[5]^m3[7];

R1[3] = m2[6]^m3[5]^m3[7];

R2[1] = m0[1]^m1[0]^m3[6];

R2[2] = m0[1]^m3[6]^m2[7];

R2[3] = m2[7]^m3[6]^m1[0];

R3[1] = m1[1]^m2[0]^m0[0];

R3[2] = m0[0]^m0[2]^m2[0];

R3[3] = m1[1]^m0[2]^m0[0];

R4[1] = m0[7]^m1[6]^m3[4];

R4[2] = m0[7]^m2[5]^m3[4];

R4[3] = m3[4]^m2[5]^m1[6];

R5[1] = m0[6]^m1[5]^m3[3];

R5[2] = m0[6]^m2[4]^m3[3];

R5[3] = m3[3]^m2[4]^m1[5];

R6[1] = m0[5]^m1[4]^m3[2];

R6[2] = m0[5]^m2[3]^m3[2];

R6[3] = m3[2]^m2[3]^m1[4];

R7[1] = m0[4]^m1[3]^m3[1];

R7[2] = m0[4]^m2[2]^m3[1];

R7[3] = m3[1]^m2[2]^m1[3];

R8[1] = m0[3]^m1[2]^m3[0];

R8[2] = m0[3]^m2[1]^m3[0];

R8[3] = m3[0]^m2[1]^m1[2];

O[55:0] = {m3[7],m3[5],m2[6],R1[3],m1[7],R1[2],R1[1],m3[6],m2[7],m1[0],R2[3],m0[1],R2[2],R2[1],m0[0],m0[2],m1[1],R3[3],m2[0],R3[2],R3[1],m3[4],m2[5],m1[6],R3[3],m2[0],R3[2],R3[1],m3[4],m2[5],m1[6],R4[3],m0[7],R4[2],R4[1],m3[3],m2[4],m1[5],R5[3],m0[6],R5[2],R5[1],m3[2],m2[3],m1[4],R6[3],m0[5],R6[2],R6[1],m3[1],m2[2],m1[3],R7[3],m0[4],R7[2],R7[1],m3[0],m2[1],m1[2],R8[3],m0[3],R8[2],R8[1]};

end

end

endmodule