module primary\_lsfr4 (

input clk,

input reset,

input write,

input pushin,

input [85:0] InitialData4,

output [85:0] rnd1

//output [8:0] rnd2,

//output [15:0] rnd3,

//output [9:0] rnd4

);

//Linear feedback shift registers

reg [85:0] lfsr4, random\_next1, random\_done1;

//Count for the number of shifts

reg [3:0] count1, count\_next1;

always @ (posedge clk or posedge reset)

begin

if (reset)

begin

lfsr4 <= #1 0;

//case1

//lfsr1 <= 185'h4751245563371bb82b2b5aacd05678a1b17e06c62eb0dace; //An LFSR cannot have an all 0 state, thus reset to 4751245563371bb82b2b5aacd05678a1b17e06c62eb0dace

end

else

begin

if (write)

begin

lfsr4 <= InitialData4;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr4 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

//-----------Combinational code for shift register 1 --> 13 bits ----------//

random\_next1 = lfsr4; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr4[71]^lfsr4[77]^lfsr4[83]) ,(lfsr4[70]^lfsr4[76]^lfsr4[82]) ,(lfsr4[69]^lfsr4[75]^lfsr4[81]) ,(lfsr4[68]^lfsr4[74]^lfsr4[80]) ,

(lfsr4[67]^lfsr4[73]^lfsr4[79]^lfsr4[85]) ,(lfsr4[66]^lfsr4[72]^lfsr4[78]^lfsr4[84]) ,(lfsr4[65:10]), (lfsr4[9]^lfsr4[85]),

(lfsr4[8]^lfsr4[84]), (lfsr4[7]^lfsr4[83]), (lfsr4[6]^lfsr4[85]^lfsr4[82]),(lfsr4[5]^lfsr4[84]^lfsr4[81]),(lfsr4[4]^lfsr4[83]^lfsr4[80]),

(lfsr4[3]^lfsr4[82]^lfsr4[79]^lfsr4[85]), (lfsr4[2]^lfsr4[81]^lfsr4[78]^lfsr4[84]), (lfsr4[1]^lfsr4[80]^lfsr4[77]^lfsr4[83]),

(lfsr4[0]^lfsr4[79]^lfsr4[85]^lfsr4[76]^lfsr4[82]), (lfsr4[85]^lfsr4[78]^lfsr4[84]^lfsr4[75]^lfsr4[81]) ,(lfsr4[84]^lfsr4[77]^lfsr4[83]^lfsr4[74]^lfsr4[80]) ,

(lfsr4[83]^lfsr4[76]^lfsr4[82]^lfsr4[73]^lfsr4[79]^lfsr4[85]) ,(lfsr4[82]^lfsr4[75]^lfsr4[81]^lfsr4[72]^lfsr4[78]^lfsr4[84]) ,

(lfsr4[81]^lfsr4[74]^lfsr4[80]) ,(lfsr4[80]^lfsr4[73]^lfsr4[79]^lfsr4[85]) ,(lfsr4[79]^lfsr4[85]^lfsr4[72]^lfsr4[78]^lfsr4[84]) ,

(lfsr4[78]^lfsr4[84]) ,(lfsr4[77]^lfsr4[83]) ,(lfsr4[76]^lfsr4[82]) ,(lfsr4[75]^lfsr4[81]) ,(lfsr4[74]^lfsr4[80]) ,

(lfsr4[73]^lfsr4[79]^lfsr4[85]) ,(lfsr4[72]^lfsr4[78]^lfsr4[84]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr4; //assign the random number to output after 13 shifts

end

//--------------------------------------------End of combination logic for shift register 1----------------------------------//

end

assign rnd1 = lfsr4;

endmodule