module primary\_lsfr3 (

input clk,

input reset,

input write,

input pushin,

input [72:0] InitialData3,

output [72:0] rnd1

//output [8:0] rnd2,

//output [15:0] rnd3,

//output [9:0] rnd4

);

//Linear feedback shift registers

reg [72:0] lfsr1, 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

lfsr1 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr1 <= InitialData3;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr1 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

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

random\_next1 = lfsr1; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr1[61]), (lfsr1[60]), (lfsr1[59]), (lfsr1[58]) , (lfsr1[57]^lfsr1[72])

,(lfsr1[56]^lfsr1[71]) ,(lfsr1[55]^lfsr1[70]) ,(lfsr1[54]^lfsr1[69]) ,(lfsr1[53]^lfsr1[68]) ,

(lfsr1[52]^lfsr1[67]) ,(lfsr1[51]^lfsr1[66]) ,(lfsr1[50]^lfsr1[65]) ,(lfsr1[49]^lfsr1[72]^lfsr1[64]) ,

(lfsr1[48]^lfsr1[71]^lfsr1[63]) ,(lfsr1[47]^lfsr1[70]^lfsr1[62]) ,(lfsr1[46]^lfsr1[69]) ,

(lfsr1[45]^lfsr1[68]) ,(lfsr1[44]^lfsr1[67]) ,(lfsr1[43]^lfsr1[66]) ,(lfsr1[42]^lfsr1[65]) ,

(lfsr1[41]^lfsr1[64]) ,(lfsr1[40]^lfsr1[63]) ,(lfsr1[39]^lfsr1[62]) ,(lfsr1[38]), (lfsr1[37]),

(lfsr1[36]), (lfsr1[35]), (lfsr1[34]), (lfsr1[33]), (lfsr1[32]), (lfsr1[31]), (lfsr1[30]), (lfsr1[29]),

(lfsr1[28]), (lfsr1[27]), (lfsr1[26]), (lfsr1[25]), (lfsr1[24]), (lfsr1[23]), (lfsr1[22]),

(lfsr1[21]), (lfsr1[20]), (lfsr1[19]), (lfsr1[18]), (lfsr1[17]), (lfsr1[16]), (lfsr1[15]),

(lfsr1[14]), (lfsr1[13]), (lfsr1[12]), (lfsr1[11]), (lfsr1[10]^lfsr1[72]), (lfsr1[9]^lfsr1[71]),

(lfsr1[8]^lfsr1[70]), (lfsr1[7]^lfsr1[69]), (lfsr1[6]^lfsr1[68]), (lfsr1[5]^lfsr1[67]),

(lfsr1[4]^lfsr1[66]), (lfsr1[3]^lfsr1[65]) ,(lfsr1[2]^lfsr1[64]) ,(lfsr1[1]^lfsr1[63]) ,

(lfsr1[0]^lfsr1[62]), (lfsr1[72]) ,(lfsr1[71]), (lfsr1[70]), (lfsr1[69]), (lfsr1[68]),

(lfsr1[67]), (lfsr1[66]), (lfsr1[65]), (lfsr1[64]), (lfsr1[63]), (lfsr1[62]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr1; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr1;

endmodule