module primary\_lsfr2 (

input clk,

input reset,

input write,

input pushin,

input [42:0] InitialData2,

output [42:0] rnd1

);

//Linear feedback shift registers

reg [42:0] lfsr2, 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

lfsr2 <= #1 0;

//case1

end

else

begin

if (write)

begin

lfsr2 <= InitialData2;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr2 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

/\*always @ (posedge clk or posedge reset)

begin

if (reset)

begin

lfsr2 <= InitialData2;

count1 <= 0;

end

else if (pushin)

begin

lfsr2 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

\*/

always @ (\*)

begin

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

random\_next1 = lfsr2; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr2[28]^lfsr2[41]^lfsr2[39]^lfsr2[36]) ,(lfsr2[27]^lfsr2[40]^lfsr2[38]^lfsr2[35]) ,(lfsr2[26]^lfsr2[39]^lfsr2[37]^lfsr2[34]^lfsr2[42]) ,

(lfsr2[25]^lfsr2[38]^lfsr2[36]^lfsr2[33]^lfsr2[41]) ,(lfsr2[24]^lfsr2[42]^lfsr2[37]^lfsr2[35]^lfsr2[32]^lfsr2[40]) ,

(lfsr2[23]^lfsr2[41]^lfsr2[36]^lfsr2[34]^lfsr2[31]^lfsr2[39]) ,(lfsr2[22]^lfsr2[40]^lfsr2[35]^lfsr2[33]^lfsr2[30]^lfsr2[38]) ,

(lfsr2[21]^lfsr2[39]^lfsr2[34]^lfsr2[32]^lfsr2[29]^lfsr2[37]) ,(lfsr2[20]^lfsr2[38]^lfsr2[33]^lfsr2[41]^lfsr2[31]^lfsr2[42]^lfsr2[39]) ,

(lfsr2[19]^lfsr2[37]^lfsr2[32]^lfsr2[40]^lfsr2[30]^lfsr2[41]^lfsr2[38]) ,(lfsr2[18]^lfsr2[36]^lfsr2[31]^lfsr2[39]^lfsr2[29]^lfsr2[40]^lfsr2[37]) ,

(lfsr2[17]^lfsr2[35]^lfsr2[30]^lfsr2[41]^lfsr2[38]) ,(lfsr2[16]^lfsr2[34]^lfsr2[29]^lfsr2[40]^lfsr2[37]) ,(lfsr2[15]^lfsr2[33]^lfsr2[41]) ,

(lfsr2[14]^lfsr2[32]^lfsr2[40]) ,(lfsr2[13]^lfsr2[31]^lfsr2[42]^lfsr2[39]) ,(lfsr2[12]^lfsr2[30]^lfsr2[41]^lfsr2[38]) ,

(lfsr2[11]^lfsr2[29]^lfsr2[42]^lfsr2[40]^lfsr2[37]) ,(lfsr2[10:8]), (lfsr2[07]^lfsr2[42]) ,(lfsr2[06]^lfsr2[41]) ,

(lfsr2[05]^lfsr2[40]) ,(lfsr2[04]^lfsr2[39]) ,(lfsr2[03]^lfsr2[38]) ,(lfsr2[02]^lfsr2[37]) ,(lfsr2[01]^lfsr2[36]) ,(lfsr2[00]^lfsr2[35]) ,

(lfsr2[34:32]), (lfsr2[31]^lfsr2[42]) ,(lfsr2[30]^lfsr2[41]) ,(lfsr2[29]^lfsr2[42]^lfsr2[40]) ,(lfsr2[36:35]), (lfsr2[34]^lfsr2[42]) ,

(lfsr2[33]^lfsr2[41]) ,(lfsr2[32]^lfsr2[40]) ,(lfsr2[31]^lfsr2[42]^lfsr2[39]) ,(lfsr2[30]^lfsr2[41]^lfsr2[38]) ,(lfsr2[29]^lfsr2[42]^lfsr2[40]^lfsr2[37]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr2; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr2;

endmodule