module primary\_lsfr0 (

input clk,

input reset,

input write,

input pushin,

input [449:0] InitialData0,

output [449:0] rnd1

);

//Linear feedback shift registers

reg [449:0] lfsr0, 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

lfsr0 <= #1 0;

end

else

begin

if (write)

begin

lfsr0 <= #1 InitialData0;

count1 <= #1 0;

end

else if (pushin)

begin

lfsr0 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

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

random\_next1 = lfsr0; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = {

(lfsr0[431:167]), (lfsr0[166]^lfsr0[449]) ,(lfsr0[165]^lfsr0[448]) ,(lfsr0[164]^lfsr0[447]) ,(lfsr0[163]^lfsr0[446]) ,(lfsr0[162]^lfsr0[445]) ,

(lfsr0[161]^lfsr0[444]) ,(lfsr0[160]^lfsr0[443]) ,(lfsr0[159]^lfsr0[442]) ,(lfsr0[158]^lfsr0[441]) ,(lfsr0[157]^lfsr0[440]) ,

(lfsr0[156]^lfsr0[439]) ,(lfsr0[155]^lfsr0[438]) ,(lfsr0[154]^lfsr0[437]) ,(lfsr0[153]^lfsr0[436]) ,(lfsr0[152]^lfsr0[435]) ,

(lfsr0[151]^lfsr0[434]) ,(lfsr0[150]^lfsr0[433]) ,(lfsr0[149]^lfsr0[432]) , (lfsr0[148:128]), (lfsr0[127]^lfsr0[449]) ,(lfsr0[126]^lfsr0[448]) ,

(lfsr0[125]^lfsr0[447]) ,(lfsr0[124]^lfsr0[446]) ,(lfsr0[123]^lfsr0[445]) ,(lfsr0[122]^lfsr0[444]) ,(lfsr0[121]^lfsr0[443]) ,

(lfsr0[120]^lfsr0[442]) ,(lfsr0[119]^lfsr0[441]) ,(lfsr0[118]^lfsr0[440]) ,(lfsr0[117]^lfsr0[439]) ,(lfsr0[116]^lfsr0[438]) ,

(lfsr0[115]^lfsr0[437]) ,(lfsr0[114]^lfsr0[436]) ,(lfsr0[113]^lfsr0[435]) ,(lfsr0[112]^lfsr0[434]) ,(lfsr0[111]^lfsr0[433]) ,

(lfsr0[110]^lfsr0[432]) ,(lfsr0[109:70]), (lfsr0[069]^lfsr0[449]) ,(lfsr0[068]^lfsr0[448]) ,(lfsr0[067]^lfsr0[447]) ,(lfsr0[066]^lfsr0[446]) ,

(lfsr0[065]^lfsr0[445]) ,(lfsr0[064]^lfsr0[444]) ,(lfsr0[063]^lfsr0[443]) ,(lfsr0[062]^lfsr0[442]) ,(lfsr0[061]^lfsr0[441]) ,

(lfsr0[060]^lfsr0[440]) ,(lfsr0[059]^lfsr0[439]) ,(lfsr0[058]^lfsr0[438]) ,(lfsr0[057]^lfsr0[437]) ,(lfsr0[056]^lfsr0[436]) ,

(lfsr0[055]^lfsr0[435]) ,(lfsr0[054]^lfsr0[434]) ,(lfsr0[053]^lfsr0[433]) ,(lfsr0[052]^lfsr0[432]) ,(lfsr0[51]), (lfsr0[50]) ,(lfsr0[049]^lfsr0[449]) ,

(lfsr0[048]^lfsr0[448]) ,(lfsr0[047]^lfsr0[447]) ,(lfsr0[046]^lfsr0[446]) ,(lfsr0[045]^lfsr0[445]) ,(lfsr0[044]^lfsr0[444]) ,

(lfsr0[043]^lfsr0[443]) ,(lfsr0[042]^lfsr0[442]) ,(lfsr0[041]^lfsr0[441]) ,(lfsr0[040]^lfsr0[440]) ,(lfsr0[039]^lfsr0[439]) ,

(lfsr0[038]^lfsr0[438]) ,(lfsr0[037]^lfsr0[437]) ,(lfsr0[036]^lfsr0[436]) ,(lfsr0[035]^lfsr0[435]) ,(lfsr0[034]^lfsr0[434]) ,

(lfsr0[033]^lfsr0[433]) ,(lfsr0[032]^lfsr0[432]) ,(lfsr0[31:13]), (lfsr0[12]^lfsr0[449]) ,(lfsr0[11]^lfsr0[448]) ,(lfsr0[10]^lfsr0[447]) ,

(lfsr0[9]^lfsr0[446]) ,(lfsr0[8]^lfsr0[445]) ,(lfsr0[7]^lfsr0[444]) ,(lfsr0[6]^lfsr0[443]) ,(lfsr0[5]^lfsr0[442]) ,

(lfsr0[4]^lfsr0[441]) ,(lfsr0[3]^lfsr0[440]) ,(lfsr0[2]^lfsr0[439]) ,(lfsr0[1]^lfsr0[438]) ,(lfsr0[0]^lfsr0[437]) ,

(lfsr0[449]^lfsr0[436]) ,(lfsr0[448]^lfsr0[435]) ,(lfsr0[447]^lfsr0[434]) ,(lfsr0[446]^lfsr0[433]) ,(lfsr0[445]^lfsr0[432]) , (lfsr0[444:432]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr0; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr0;

endmodule