module primary\_lsfr11 (

input clk,

input reset,

input write,

input pushin,

input [52:0] InitialData11,

output [52:0] rnd1

);

//Linear feedback shift registers

reg [52:0] lfsr11, 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

lfsr11 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr11 <= InitialData11;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr11 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

/\*

always @ (posedge clk or posedge reset)

begin

if (reset)

begin

//lfsr11 <= 53'hb8dafe28a7b33e12; //An LFSR cannot have an all 0 state, thus reset to b8dafe28a7b33e12

lfsr11 <= InitialData11;

count1 <= 0;

end

else if (pushin)

begin

lfsr11 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

\*/

always @ (\*)

begin

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

random\_next1 = lfsr11; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr11[39]^lfsr11[41]^lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[38]^lfsr11[40]^lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,(lfsr11[37]), (lfsr11[36]^lfsr11[52]) ,

(lfsr11[35]^lfsr11[51]) ,(lfsr11[34]^lfsr11[50]^lfsr11[52]) ,(lfsr11[33]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[32]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,(lfsr11[31]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[30]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,(lfsr11[29]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[28]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]) ,(lfsr11[27]^lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]) ,

(lfsr11[26]^lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]) ,(lfsr11[25]^lfsr11[41]^lfsr11[43]^lfsr11[45]^lfsr11[47]) ,

(lfsr11[24]^lfsr11[40]^lfsr11[42]^lfsr11[44]^lfsr11[46]) ,(lfsr11[23]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[22]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,(lfsr11[21]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[20]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,(lfsr11[19]^lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[18]^lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[17]^lfsr11[52]^lfsr11[41]^lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[16]^lfsr11[51]^lfsr11[40]^lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[15]^lfsr11[50]^lfsr11[52]) ,(lfsr11[14]^lfsr11[49]^lfsr11[51]) ,(lfsr11[13]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[12]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,(lfsr11[11]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[10]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,(lfsr11[09]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[08]^lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,(lfsr11[07]^lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[06]^lfsr11[41]^lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,

(lfsr11[05]^lfsr11[40]^lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[04]), (lfsr11[03]^lfsr11[52]) ,(lfsr11[02]^lfsr11[51]) ,(lfsr11[01]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[00]^lfsr11[49]^lfsr11[51]) ,(lfsr11[48]^lfsr11[50]) ,(lfsr11[47]^lfsr11[49]) ,(lfsr11[46]^lfsr11[48]) ,

(lfsr11[45]^lfsr11[47]) ,(lfsr11[44]^lfsr11[46]) ,(lfsr11[43]^lfsr11[45]) ,(lfsr11[42]^lfsr11[44]) ,(lfsr11[41]^lfsr11[43]) ,

(lfsr11[40]^lfsr11[42]) ,(lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,(lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) ,

(lfsr11[41]^lfsr11[43]^lfsr11[45]^lfsr11[47]^lfsr11[49]^lfsr11[51]) ,(lfsr11[40]^lfsr11[42]^lfsr11[44]^lfsr11[46]^lfsr11[48]^lfsr11[50]^lfsr11[52]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr11; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr11;

endmodule