module primary\_lsfr10 (

input clk,

input reset,

input write,

input pushin,

input [49:0] InitialData10,

output [49:0] rnd1

);

//Linear feedback shift registers

reg [49:0] lfsr10, 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

lfsr10 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr10 <= InitialData10;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr10 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

/\*

always @ (posedge clk or posedge reset)

begin

if (reset)

begin

//lfsr10 <= 50'hf1b91c2fb6dddfa9; //An LFSR cannot have an all 0 state, thus reset to f1b91c2fb6dddfa9

lfsr10 <= InitialData10;

count1 <= 0;

end

else if (pushin)

begin

lfsr10 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

\*/

always @ (\*)

begin

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

random\_next1 = lfsr10; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr10[37:36]), (lfsr10[35]^lfsr10[49]) ,(lfsr10[34]^lfsr10[48]) ,(lfsr10[33]^lfsr10[47]) ,(lfsr10[32]^lfsr10[46]) ,

(lfsr10[31]^lfsr10[45]) ,(lfsr10[30]^lfsr10[44]) ,(lfsr10[29]^lfsr10[43]) ,(lfsr10[28]^lfsr10[42]) ,(lfsr10[27]^lfsr10[41]) ,

(lfsr10[26]^lfsr10[40]) ,(lfsr10[25]^lfsr10[39]) ,(lfsr10[24]^lfsr10[38]) , (lfsr10[23:21]), (lfsr10[20]^lfsr10[49]) ,

(lfsr10[19]^lfsr10[48]) ,(lfsr10[18]^lfsr10[47]) ,(lfsr10[17]^lfsr10[46]) ,(lfsr10[16]^lfsr10[45]) ,(lfsr10[15]^lfsr10[49]^lfsr10[44]) ,

(lfsr10[14]^lfsr10[48]^lfsr10[43]) ,(lfsr10[13]^lfsr10[47]^lfsr10[42]) ,(lfsr10[12]^lfsr10[46]^lfsr10[41]) ,

(lfsr10[11]^lfsr10[45]^lfsr10[40]) ,(lfsr10[10]^lfsr10[44]^lfsr10[39]) ,(lfsr10[09]^lfsr10[43]^lfsr10[38]) ,(lfsr10[08]^lfsr10[42]) ,

(lfsr10[07]^lfsr10[41]) ,(lfsr10[06]^lfsr10[40]) ,(lfsr10[05]^lfsr10[49]^lfsr10[39]) ,(lfsr10[04]^lfsr10[49]^lfsr10[48]^lfsr10[38]) ,

(lfsr10[03]^lfsr10[48]^lfsr10[47]) ,(lfsr10[02]^lfsr10[47]^lfsr10[46]) ,(lfsr10[01]^lfsr10[46]^lfsr10[45]) ,

(lfsr10[00]^lfsr10[45]^lfsr10[44]) ,(lfsr10[49]^lfsr10[44]^lfsr10[43]) ,(lfsr10[48]^lfsr10[43]^lfsr10[42]) ,

(lfsr10[47]^lfsr10[42]^lfsr10[41]) ,(lfsr10[46]^lfsr10[41]^lfsr10[40]) ,(lfsr10[45]^lfsr10[40]^lfsr10[39]) ,

(lfsr10[44]^lfsr10[39]^lfsr10[38]) ,(lfsr10[43]^lfsr10[38]) ,(lfsr10[42:38]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr10; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr10;

endmodule