module primary\_lsfr5 (

input clk,

input reset,

input pushin,

input write,

input [300:0] InitialData5,

output [300:0] rnd1

);

//Linear feedback shift registers

reg [300:0] lfsr5, 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

lfsr5 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr5 <= InitialData5;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr5 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

//if (pushin)

//begin

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

random\_next1 = lfsr5; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr5[286:277]), (lfsr5[300]^lfsr5[276]) ,(lfsr5[299]^lfsr5[275]) ,(lfsr5[298]^lfsr5[274]) ,(lfsr5[297]^lfsr5[273]) ,(lfsr5[296]^lfsr5[272]) ,

(lfsr5[295]^lfsr5[271]) ,(lfsr5[294]^lfsr5[270]) ,(lfsr5[293]^lfsr5[269]) ,(lfsr5[292]^lfsr5[268]) ,(lfsr5[291]^lfsr5[267]) ,

(lfsr5[290]^lfsr5[266]) ,(lfsr5[289]^lfsr5[265]) ,(lfsr5[288]^lfsr5[264]) ,(lfsr5[287]^lfsr5[263]) , (lfsr5[262:222]), (lfsr5[300]^lfsr5[221]) ,

(lfsr5[299]^lfsr5[220]) ,(lfsr5[298]^lfsr5[219]) ,(lfsr5[297]^lfsr5[218]) ,(lfsr5[296]^lfsr5[217]) ,(lfsr5[295]^lfsr5[216]) ,

(lfsr5[294]^lfsr5[215]) ,(lfsr5[293]^lfsr5[214]) ,(lfsr5[292]^lfsr5[213]) ,(lfsr5[291]^lfsr5[212]) ,(lfsr5[290]^lfsr5[211]) ,

(lfsr5[289]^lfsr5[210]) , (lfsr5[288]^lfsr5[300]^lfsr5[209]) , (lfsr5[287]^lfsr5[299]^lfsr5[208]), (lfsr5[298]^lfsr5[207]) ,(lfsr5[297]^lfsr5[206]) ,(lfsr5[296]^lfsr5[205]) ,(lfsr5[295]^lfsr5[204]) ,

(lfsr5[294]^lfsr5[203]) ,(lfsr5[293]^lfsr5[202]) ,(lfsr5[292]^lfsr5[201]) ,(lfsr5[291]^lfsr5[200]) ,(lfsr5[290]^lfsr5[199]) ,

(lfsr5[289]^lfsr5[198]) ,(lfsr5[288]^lfsr5[197]) ,(lfsr5[287]^lfsr5[196]) ,(lfsr5[300]^lfsr5[195]) ,(lfsr5[299]^lfsr5[194]) ,

(lfsr5[298]^lfsr5[193]) ,(lfsr5[297]^lfsr5[192]) ,(lfsr5[296]^lfsr5[191]) ,(lfsr5[295]^lfsr5[190]) ,(lfsr5[294]^lfsr5[189]) ,

(lfsr5[293]^lfsr5[188]) ,(lfsr5[292]^lfsr5[187]) ,(lfsr5[291]^lfsr5[186]) ,(lfsr5[290]^lfsr5[185]) ,(lfsr5[289]^lfsr5[184]) ,

(lfsr5[288]^lfsr5[183]) ,(lfsr5[287]^lfsr5[182]) , (lfsr5[181:33]), (lfsr5[300]^lfsr5[32]) ,(lfsr5[299]^lfsr5[31]) ,(lfsr5[298]^lfsr5[30]) ,

(lfsr5[297]^lfsr5[29]) ,(lfsr5[296]^lfsr5[28]) ,(lfsr5[295]^lfsr5[27]) ,(lfsr5[294]^lfsr5[26]) ,(lfsr5[293]^lfsr5[25]) ,

(lfsr5[292]^lfsr5[24]) ,(lfsr5[291]^lfsr5[23]) ,(lfsr5[290]^lfsr5[22]) ,(lfsr5[289]^lfsr5[21]) ,(lfsr5[288]^lfsr5[20]) ,

(lfsr5[287]^lfsr5[19]), (lfsr5[18:0]), (lfsr5[300:287]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr5; //assign the random number to output after 13 shifts

end

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

end

//end

assign rnd1 = lfsr5;

endmodule