module primary\_lsfr8 (

input clk,

input reset,

input write,

input pushin,

input [69:0] InitialData8,

output [69:0] rnd1

);

//Linear feedback shift registers

reg [69:0] lfsr8, 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

lfsr8 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr8 <= InitialData8;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr8 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

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

random\_next1 = lfsr8; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr8[56]^lfsr8[68]^lfsr8[63]) ,(lfsr8[55]^lfsr8[67]^lfsr8[62]^lfsr8[69]) ,(lfsr8[54]^lfsr8[66]^lfsr8[61]^lfsr8[68]) ,(lfsr8[53]^lfsr8[65]^lfsr8[60]^lfsr8[67]) ,

(lfsr8[52]^lfsr8[64]^lfsr8[59]^lfsr8[66]) ,(lfsr8[51]^lfsr8[63]^lfsr8[58]^lfsr8[65]) ,(lfsr8[50]^lfsr8[62]^lfsr8[57]^lfsr8[64]) ,(lfsr8[49]^lfsr8[61]^lfsr8[68]) ,

(lfsr8[48]^lfsr8[60]^lfsr8[67]) ,(lfsr8[47]^lfsr8[59]^lfsr8[66]) ,(lfsr8[46]^lfsr8[58]^lfsr8[65]) ,(lfsr8[45]^lfsr8[57]^lfsr8[69]^lfsr8[64]) ,(lfsr8[44]), (lfsr8[43]),

(lfsr8[42]^lfsr8[69]) ,(lfsr8[41]^lfsr8[68]) ,(lfsr8[40]^lfsr8[67]) ,(lfsr8[39]^lfsr8[66]) ,(lfsr8[38]^lfsr8[65]) ,(lfsr8[37]^lfsr8[64]) ,(lfsr8[36]^lfsr8[63]) ,

(lfsr8[35]^lfsr8[62]^lfsr8[69]) ,(lfsr8[34]^lfsr8[61]^lfsr8[68]) ,(lfsr8[33]^lfsr8[69]^lfsr8[60]^lfsr8[67]) ,(lfsr8[32]^lfsr8[68]^lfsr8[59]^lfsr8[66]) ,(lfsr8[31]^lfsr8[67]^lfsr8[58]^lfsr8[65]) ,

(lfsr8[30]^lfsr8[66]^lfsr8[57]^lfsr8[69]^lfsr8[64]) ,(lfsr8[29]^lfsr8[69]^lfsr8[65]) ,(lfsr8[28]^lfsr8[68]^lfsr8[64]) ,(lfsr8[27]^lfsr8[67]^lfsr8[63]) ,(lfsr8[26]^lfsr8[66]^lfsr8[62]^lfsr8[69]) ,

(lfsr8[25]^lfsr8[65]^lfsr8[61]^lfsr8[68]) ,(lfsr8[24]^lfsr8[64]^lfsr8[60]^lfsr8[67]) ,(lfsr8[23]^lfsr8[63]^lfsr8[59]^lfsr8[66]) ,(lfsr8[22]^lfsr8[62]^lfsr8[69]^lfsr8[58]^lfsr8[65]) ,

(lfsr8[21]^lfsr8[61]^lfsr8[57]^lfsr8[64]^lfsr8[68]^lfsr8[69]) ,(lfsr8[20]^lfsr8[60]^lfsr8[67]) ,(lfsr8[19]^lfsr8[59]^lfsr8[66]) ,(lfsr8[18]^lfsr8[58]^lfsr8[65]) ,

(lfsr8[17]^lfsr8[57]^lfsr8[69]^lfsr8[64]) ,(lfsr8[16:0]), (lfsr8[69:63]), (lfsr8[62]^lfsr8[69]) ,(lfsr8[61]^lfsr8[68]) ,(lfsr8[60]^lfsr8[67]) ,(lfsr8[59]^lfsr8[66]) ,(lfsr8[58]^lfsr8[65]) ,

(lfsr8[57]^lfsr8[69]^lfsr8[64]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr8; //assign the random number to output after 13 shifts

end

end

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

assign rnd1 = lfsr8;

endmodule