module primary\_lsfr14 (

input clk,

input reset,

input write,

input pushin,

input [63:0] InitialData14,

output [63:0] rnd1

);

//Linear feedback shift registers

reg [63:0] lfsr14, 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

lfsr14 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr14 <= InitialData14;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr14 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

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

random\_next1 = lfsr14; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr14[49]^lfsr14[57]^lfsr14[60]^lfsr14[63]^lfsr14[52]^lfsr14[60]^lfsr14[63]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,

(lfsr14[48]^lfsr14[56]^lfsr14[59]^lfsr14[62]^lfsr14[51]^lfsr14[59]^lfsr14[62]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[47]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]^lfsr14[50]^lfsr14[58]^lfsr14[61]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[46]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,(lfsr14[45]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[44]^lfsr14[52]^lfsr14[60]^lfsr14[63]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,

(lfsr14[43]^lfsr14[51]^lfsr14[59]^lfsr14[62]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[42]^lfsr14[50]^lfsr14[58]^lfsr14[61]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[41:31]), (lfsr14[30]^lfsr14[63]), (lfsr14[29]^lfsr14[62]) ,(lfsr14[28]^lfsr14[61]) ,(lfsr14[27]^lfsr14[63]^lfsr14[60]^lfsr14[63]) ,(lfsr14[26]^lfsr14[62]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[25]^lfsr14[61]^lfsr14[58]^lfsr14[61]) ,(lfsr14[24]^lfsr14[60]^lfsr14[63]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[23]^lfsr14[59]^lfsr14[62]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[22]^lfsr14[63]^lfsr14[58]^lfsr14[61]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,

(lfsr14[21]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[20]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[19]^lfsr14[60]^lfsr14[63]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]^lfsr14[52]^lfsr14[60]^lfsr14[63]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,

(lfsr14[18]^lfsr14[59]^lfsr14[62]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]^lfsr14[51]^lfsr14[59]^lfsr14[62]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[17]^lfsr14[58]^lfsr14[61]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]^lfsr14[50]^lfsr14[58]^lfsr14[61]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[16]^lfsr14[57]^lfsr14[60]^lfsr14[63]^lfsr14[52]^lfsr14[60]^lfsr14[63]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,

(lfsr14[15]^lfsr14[56]^lfsr14[59]^lfsr14[62]^lfsr14[51]^lfsr14[59]^lfsr14[62]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[14]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]^lfsr14[50]^lfsr14[58]^lfsr14[61]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[13]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,(lfsr14[12]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[11]^lfsr14[52]^lfsr14[60]^lfsr14[63]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,

(lfsr14[10]^lfsr14[51]^lfsr14[59]^lfsr14[62]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[09]^lfsr14[50]^lfsr14[58]^lfsr14[61]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,

(lfsr14[8:0]) , (lfsr14[63:61]),

(lfsr14[60]^lfsr14[63]) ,(lfsr14[59]^lfsr14[62]) ,(lfsr14[58]^lfsr14[61]) ,(lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[56]^lfsr14[59]^lfsr14[62]) ,(lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,(lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) ,(lfsr14[52]^lfsr14[60]^lfsr14[63]^lfsr14[55]^lfsr14[63]^lfsr14[58]^lfsr14[61]) ,

(lfsr14[51]^lfsr14[59]^lfsr14[62]^lfsr14[54]^lfsr14[62]^lfsr14[57]^lfsr14[60]^lfsr14[63]) ,

(lfsr14[50]^lfsr14[58]^lfsr14[61]^lfsr14[53]^lfsr14[61]^lfsr14[56]^lfsr14[59]^lfsr14[62]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr14; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr14;

endmodule