module primary\_lsfr9 (

input clk,

input reset,

input write,

input pushin,

input [126:0] InitialData9,

output [126:0] rnd1

);

//Linear feedback shift registers

reg [126:0] lfsr9, 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

lfsr9 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr9 <= InitialData9;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr9 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

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

random\_next1 = lfsr9; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr9[110:54]), (lfsr9[53]^lfsr9[126]), (lfsr9[52]^lfsr9[125]), (lfsr9[51]^lfsr9[124]), (lfsr9[50]^lfsr9[123]), (lfsr9[49]^lfsr9[122]),

(lfsr9[48]^lfsr9[121]), (lfsr9[47]^lfsr9[120]), (lfsr9[46]^lfsr9[119]), (lfsr9[45]^lfsr9[118]),

(lfsr9[44]^lfsr9[126]^lfsr9[117]), (lfsr9[43]^lfsr9[125]^lfsr9[116]), (lfsr9[42]^lfsr9[124]^lfsr9[115]), (lfsr9[41]^lfsr9[123]^lfsr9[114]),

(lfsr9[40]^lfsr9[122]^lfsr9[113]), (lfsr9[39]^lfsr9[121]^lfsr9[112]), (lfsr9[38]^lfsr9[120]^lfsr9[111]), (lfsr9[37]^lfsr9[119]),

(lfsr9[36]^lfsr9[118]), (lfsr9[35]^lfsr9[117]), (lfsr9[34]^lfsr9[116]), (lfsr9[33]^lfsr9[115]), (lfsr9[32]^lfsr9[114]),

(lfsr9[31]^lfsr9[113]), (lfsr9[30]^lfsr9[112]), (lfsr9[29]^lfsr9[111]), (lfsr9[28:13]), (lfsr9[126]^lfsr9[12]), (lfsr9[125]^lfsr9[11]),

(lfsr9[124]^lfsr9[10]), (lfsr9[123]^lfsr9[9]) ,(lfsr9[122]^lfsr9[8]) , (lfsr9[121]^lfsr9[7]) , (lfsr9[120]^lfsr9[6]) ,

(lfsr9[119]^lfsr9[5]) ,(lfsr9[118]^lfsr9[4]) ,(lfsr9[117]^lfsr9[3]) ,(lfsr9[116]^lfsr9[2]) ,(lfsr9[115]^lfsr9[1]) ,(lfsr9[114]^lfsr9[0]) ,

(lfsr9[113]^lfsr9[126]) ,(lfsr9[112]^lfsr9[125]) , (lfsr9[111]^lfsr9[124]) , (lfsr9[123:111]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr9; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr9;

endmodule