module primary\_lsfr6 (

input clk,

input reset,

input write,

input pushin,

input [527:0] InitialData6,

output [527:0] rnd1

);

//Linear feedback shift registers

reg [527:0] lfsr6, 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

lfsr6 <= #1 0;

//case1

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

end

else

begin

if (write)

begin

lfsr6 <= InitialData6;

//case2

//lfsr1 <= 185'h08AAC66E37215874F559A0ACF14362FC0D24CD61E1D5512;

count1 <= 0;

end

else if (pushin)

begin

lfsr6 <= #1 random\_next1;

count1 <= #1 count\_next1;

end

end

end

always @ (\*)

begin

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

random\_next1 = lfsr6; //default state stays the same

count\_next1 = count1;

random\_done1 = 0;

random\_next1 = { (lfsr6[527]^lfsr6[517]) ,(lfsr6[526]^lfsr6[516]) ,(lfsr6[525]^lfsr6[515]) ,(lfsr6[524]^lfsr6[514]) ,(lfsr6[523]^lfsr6[513]) ,(lfsr6[522]^lfsr6[512]) ,(lfsr6[521]^lfsr6[511]) ,(lfsr6[520]^lfsr6[510]) ,

(lfsr6[519]^lfsr6[509]) ,(lfsr6[518]^lfsr6[508]) ,(lfsr6[507:442]), (lfsr6[527]^lfsr6[441]) ,(lfsr6[526]^lfsr6[440]) ,(lfsr6[525]^lfsr6[439]) ,(lfsr6[524]^lfsr6[438]) ,(lfsr6[523]^lfsr6[437]) ,(lfsr6[522]^lfsr6[436]) ,

(lfsr6[521]^lfsr6[435]) ,(lfsr6[520]^lfsr6[434]) ,(lfsr6[519]^lfsr6[433]) ,(lfsr6[518]^lfsr6[432]) ,(lfsr6[431:413]), (lfsr6[527]^lfsr6[412]) ,(lfsr6[526]^lfsr6[411]) ,(lfsr6[525]^lfsr6[410]) ,(lfsr6[524]^lfsr6[409]) ,

(lfsr6[523]^lfsr6[408]) ,(lfsr6[522]^lfsr6[407]) ,(lfsr6[521]^lfsr6[406]) ,(lfsr6[520]^lfsr6[405]) ,(lfsr6[519]^lfsr6[404]) ,(lfsr6[518]^lfsr6[403]) , (lfsr6[402:400]), (lfsr6[527]^lfsr6[399]) ,(lfsr6[526]^lfsr6[398]) ,

(lfsr6[525]^lfsr6[397]) ,(lfsr6[524]^lfsr6[396]) ,(lfsr6[523]^lfsr6[395]) ,(lfsr6[522]^lfsr6[394]) ,(lfsr6[521]^lfsr6[393]) ,(lfsr6[520]^lfsr6[392]) ,(lfsr6[519]^lfsr6[391]) ,(lfsr6[518]^lfsr6[390]) , (lfsr6[389:126]),

(lfsr6[527]^lfsr6[125]) ,(lfsr6[526]^lfsr6[124]) ,(lfsr6[525]^lfsr6[123]) ,(lfsr6[524]^lfsr6[122]) ,(lfsr6[523]^lfsr6[121]) ,(lfsr6[522]^lfsr6[120]) ,(lfsr6[521]^lfsr6[119]) ,(lfsr6[520]^lfsr6[118]) ,

(lfsr6[519]^lfsr6[117]) ,(lfsr6[518]^lfsr6[116]) , (lfsr6[115:0]), (lfsr6[527:518]) };

count\_next1 = count1 + 1;

if (count1 == 1)

begin

count1 = 0;

random\_done1 = lfsr6; //assign the random number to output after 13 shifts

end

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

end

assign rnd1 = lfsr6;

endmodule