```
//-----------------------------------------------------
// Design Name : TOP
// File Name   : lsfr_testbench.v
// Function    : Find input to LSFR that provides
//               max out combination
//-----------------------------------------------------
module top();
// -------    variables --------------
parameter WIDTH= 4,
          WIDTH2=3,
          LANES=5,
          LOOP_MAX=6'b100000;
 reg clk;
 reg reset;
 reg enable;
 integer j;
 integer combinations;

//-------- initial values for each LSFR
reg [WIDTH:0]   INIT_0;
reg [WIDTH:0]   INIT_1;
reg [WIDTH:0]   INIT_2;
reg [WIDTH:0]   INIT_3;
reg [WIDTH:0]   INIT_4;

// store number of time a case happend
reg [LANES:0] ocurrence [(2**LANES)-1:0];

// store count of LSFR only for visual purpose not really needed
wire    [WIDTH-1 : 0] lsfr1;
wire    [WIDTH-1 : 0] lsfr2;
wire    [WIDTH-1 : 0] lsfr3;
wire    [WIDTH2-1 : 0] lsfr4;
wire    [WIDTH2-1 : 0] lsfr5;
// Stores the concatenated value of the
// single bit output of
wire [LANES-1 : 0] pattern;


  initial begin
  clk = 0;
// ---------7 lines immediately below: only used if for loops are commented to assig
n preset initial values
  reset = 1;
  enable = 0;
  INIT_0=4'b0000;
  INIT_1=4'b0000;
  INIT_2=4'b0000;
  INIT_3=3'b0000;
  INIT_4=3'b000;

//------------------------        Code to generate unique input values--------------
-----------------


                                    #1 reset = 1;    // reset ensures init value
is stored
                                    #1 enable = 0;  // ensure it wont change uti
l initialization is complete

                                    // set all occurence to 0
                                    for (j=0; j< 2**LANES ; j=j+1) begin
                                    ocurrence[j]=0;
                                    end //end for loop
                                    // enables signals
                                     #1 combinations=0;
                                     #1 reset = 0;
                                     #1 enable = 1;

                                    // once done check number of unique combinat
ions genarated
                                    // Note: clock period is 2
                                     #1000  for (j=0; j< 2**LANES ; j=j+1) begin
```

```
                                                    if (ocurrence[j]>0)
                                                    combinations=combinations+1;
                                                    end //end for loop

                                                // display only if it found more than 24
                                                #1 if (combinations>=24)begin
                                                    $display ("\n %d different combinati
ons with %d, %d, %d, %d, %d\n\n",combinations,INIT_0,INIT_1,INIT_2,INIT_3,INIT_4);
                                                    end //else begin
        for (j=0; j< 2**LANES ; j=j+1) begin
         $display("pattern %d \tappeared %d \n",j, ocurrence[j] );
                end //end for loop
  #1 $finish;
end


//----------------------------   clock  ---------------------------
always #1 clk = ~clk;


//---------------------    module instantiation -------------------

sig_gen  U1(
.lout   (pattern)          ,  // Output of the counter
.enable (enable)         ,  // Enable  for counter
.clock  (clk)          ,  // clock input
.reset  (reset) ,          // reset input
.count0( lsfr1) ,
.count1(lsfr2 ) ,
.count2(lsfr3 ) ,
.count3(lsfr4 ) ,
.count4(lsfr5 ),
.INIT_0(INIT_0[WIDTH-1:0]),
.INIT_1(INIT_1[WIDTH-1:0]),
.INIT_2(INIT_2[WIDTH-1:0]),
.INIT_3(INIT_3[WIDTH2-1:0]),
.INIT_4(INIT_4[WIDTH2-1:0])
);


//------------------------  keeps track of times a pattern appears  ------------

// at neg edge keeps track of how many
// times each pattern occured
always @(negedge clk) begin
if(pattern<32)
 ocurrence[pattern]= ocurrence[pattern]+1;
end //end of always statement


endmodule
```