به نام خدا

درس: طراحی سیستم های دیجیتال

نیم‌سال تحصیلی دوم 9900

مدرس: بهاروند

تمرین شماره: 4		موعد تحویل: 1400/02/20 ساعت ۲۳:55	
نام و نام خانوادگی:	سارا آذرنوش	شماره دانشجویی:	98170668

راه حل در همین فایل ارائه شود.

فایل به PDF تبدیل و در سایت بارگذاری شود.

عنوان تمرین

یک مدار LFSR با مشخصات زیر طراحی کنید:

1) شیفت رجیستر به صورت پارامتری طراحی شود. یعنی تعداد بیتهای آن توسط پارامتری در ابتدای ماجول تعریف شود.
2) شیفت به سمت راست صورت می‌گیرد.
3) بیت کم ارزش آن دارای اندیس 1 و بیت پر ارزش آن دارای ارزش n است که n تعداد بیتهای شیفت رجیستر است.
4) با استفاده از generate کد را به نحوی بنویسید که بتواند دو چند جمله ای زیر را بسته به مقدار n پیاده سازی نماید. مقدار n نمایانگر نوع و تعداد بیتهای LFSR است.

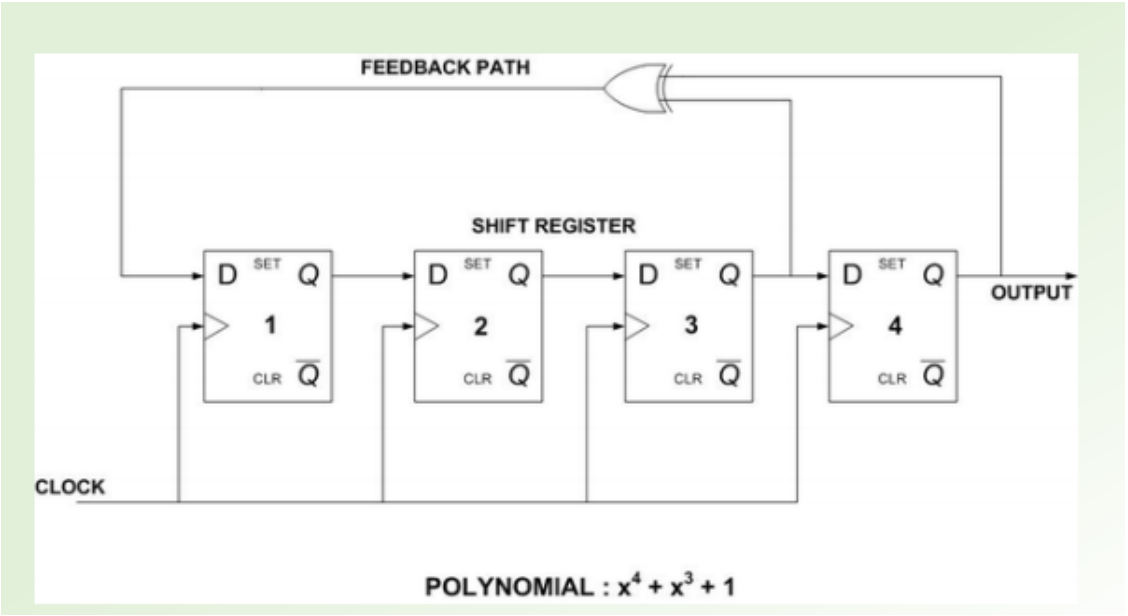
$$x^8 + x^6 + x^5 + x^1 + 1$$

$$x^{17} + x^3 + 1$$

5) برای LFSR یک ری‌ست آسنکرون به صورت active low در نظر بگیرید.
6) پریود کلاک را 10 در نظر بگیرید و کلاک active high است.
7) در testbench مقدار اولیه LFSR با استفاده سیگنال load_seed وبه صورت تصادفی (random) بارگذاری شود.
8) در testbench مقادیر محتوای LFSR را پس از بارگذاری مقدار seed و شروع به کار مدار، چاپ کنید. خروجی تقریباً شبیه شکل زیر مد نظر است. خروجی یک بیتی LFSR در این تمرین، همان بیت کم ارزش (سمت راست) است.

```
4  ----------------------------------------------------
5  LFSR WIDTH =   17
6  The current module name is = lfsr_test
7  Clock period is: 10 in the mentioned time unit
8  Clock frequency is: 100 MHz
9  Simulation started...
10
11 @    0.0000 ns:  feedback = 0,  lfsr_reg = 00000000000000000 , lfsr_out = 0,
12 @   35.0000 ns:  feedback = 0,  lfsr_reg = 00000000000000000 , lfsr_out = 0,
13 @   55.0000 ns:  feedback = 1,  lfsr_reg = 10000000000000000 , lfsr_out = 0,
14 @   65.0000 ns:  feedback = 1,  lfsr_reg = 11000000000000000 , lfsr_out = 0,
15 @   75.0000 ns:  feedback = 0,  lfsr_reg = 01100000000000000 , lfsr_out = 0,
16 @   85.0000 ns:  feedback = 0,  lfsr_reg = 00110000000000000 , lfsr_out = 0,
17 @   95.0000 ns:  feedback = 1,  lfsr_reg = 10011000000000000 , lfsr_out = 0,
18 @  105.0000 ns:  feedback = 1,  lfsr_reg = 11001100000000000 , lfsr_out = 0,
19 @  115.0000 ns:  feedback = 1,  lfsr_reg = 11100110000000000 , lfsr_out = 0,
20 @  125.0000 ns:  feedback = 0,  lfsr_reg = 01110011000000000 , lfsr_out = 0,
21 @  135.0000 ns:  feedback = 0,  lfsr_reg = 00111001100000000 , lfsr_out = 0,
22 @  145.0000 ns:  feedback = 1,  lfsr_reg = 10011100110000000 , lfsr_out = 0,
23 @  155.0000 ns:  feedback = 0,  lfsr_reg = 01001110011000000 , lfsr_out = 0,
24 @  165.0000 ns:  feedback = 0,  lfsr_reg = 00100111001100000 , lfsr_out = 0,
25 @  175.0000 ns:  feedback = 0,  lfsr_reg = 00010011100110000 , lfsr_out = 0,
26 @  185.0000 ns:  feedback = 0,  lfsr_reg = 00001001110011000 , lfsr_out = 0,
27 @  195.0000 ns:  feedback = 1,  lfsr_reg = 00000100111001100 , lfsr_out = 0,
28 @  205.0000 ns:  feedback = 0,  lfsr_reg = 10000100111001110 , lfsr_out = 0,
29 @  215.0000 ns:  feedback = 0,  lfsr_reg = 01000010011100111 , lfsr_out = 1,
30 @  225.0000 ns:  feedback = 1,  lfsr_reg = 10100000100111001 , lfsr_out = 1,
31 @  235.0000 ns:  feedback = 1,  lfsr_reg = 01010000010011100 , lfsr_out = 0,
32 @  245.0000 ns:  feedback = 0,  lfsr_reg = 10101000001001110 , lfsr_out = 0,
33 @  255.0000 ns:  feedback = 1,  lfsr_reg = 01010100000100111 , lfsr_out = 1,
34 @  265.0000 ns:  feedback = 0,  lfsr_reg = 00101010000010011 , lfsr_out = 1,
35 @  275.0000 ns:  feedback = 1,  lfsr_reg = 10010101000001001 , lfsr_out = 1,
36 @  285.0000 ns:  feedback = 1,  lfsr_reg = 01001010100000100 , lfsr_out = 0,
37 @  295.0000 ns:  feedback = 0,  lfsr_reg = 00100101010000010 , lfsr_out = 0,
```

Lfsr نوعی شیفت دهنده است که در هر کلاک از یک بیت به بیت پرارزش تر انتقال میدهد از n فلیپ فلاپ تشکیل شده است و با توجه به شماره درجه چندجمله‌ای آن را xor میکند.



در اینجا با توجه به شماره بیشترین درجه چندجمله ای اگر تعداد بیت 17 یا 8 باشد با توجه به عدد آن شیفت انجام میدهد.

در اینجا بیت کم ارزش 1 و پر ارزش n است بنابراین باید برعکس باشد و مانند شماره جدول زیر بیت ها xor میشوند.

$$x^8 + x^6 + x^5 + x + 1$$

1	2	3	4	5	6	7	8
8	7	6	5	4	3	2	1

$$x^{17} + x^3 + 1$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

*** با توجه به گفته استاد در کلاس پر ارزش n است و کم ارزش 1 بنابراین همان ضرایب چندجمله را xor میکنیم.

(code)

```
module LFSR #(parameter n = 17)(
        input clk,
        input rst,
        input load_seed,
        input [n:1]rand,
        output reg [n:1]num
        );
        wire feedback;


//x^8 + x^6 + x^5 + x^1 + 1
//x^17 + x^1 + 1
        generate
                if(n == 8 || n == 17)begin
                        if (n == 8) begin
                                assign feedback = num[8] ^ num[6]^ num[5]^ num[1];
                        end
                        else if (n == 17) begin
                          assign feedback = num[3] ^ num[17];
                        end

                        always @(posedge clk or negedge rst)begin
                                if(~rst) begin
                                        num <= 0;
                                end
                                else if(load_seed)begin
                                        num <= rand;
                                end
                                else begin
                                        num[n] <= feedback;
                                        num[n-1:1] <= num[n:2];
                                end
                        end
                end
        endgenerate
endmodule
```

اگر تعداد بیت‌ها 8 یا 17 بود انجام شود.

با توجه به تعداد بیت بیت های گفته شده را xor میکند.

اگر rst=0 بود مقدار 0 شود.

اگر load بود مقدار رندم وارد کند.

در غیر این صورت فیدبک به بیتn وارد و بیت 2 تا n به 1 تا n-1 وارد میشود.

(test)

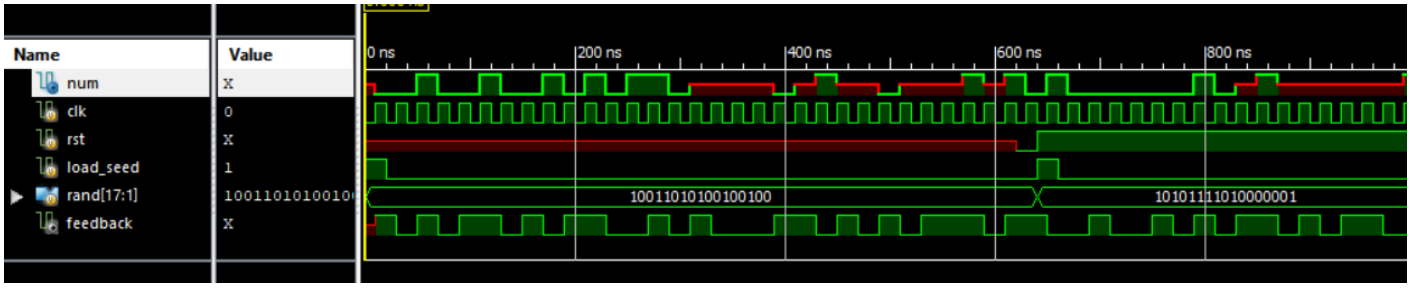
```
module test;
   reg clk;
            reg rst;
            reg load_seed;
            reg [17:1]rand;
   wire num;


   LFSR #(8) lfsr8(.num(num),.load_seed(load_seed),.clk(clk),.rand(rand));
   LFSR #(17) lfsr17(.num(num),.load_seed(load_seed),.clk(clk),.rand(rand));


 initial begin
      clk = 0;
      rand = $random;
      $monitor("%gns   feedback = %d lfsr_reg = %b  lfsr_out = %d", $time, test.lfsr17.feedback, test.lfsr17.num,
test.lfsr17.num[1]);
                        load_seed = 1;
                           #20 load_seed = 0;
                        #600
                        rst =0;
                        #20 rst = 1;
                        rand = $random;
                        $monitor("%gns   feedback = %d lfsr_reg = %b  lfsr_out = %d", $time, test.lfsr8.feedback, test.lfsr8.num,
test.lfsr8.num[1]);
                        load_seed = 1;
                         #20 load_seed = 0;
   end
      always #(10) clk = ~clk;
endmodule
```

برای تست کردن از هر دو معادله یک اینستنس ساخته و در مانیتور مانند آنچه خواسته شده نمایش میدهیم.

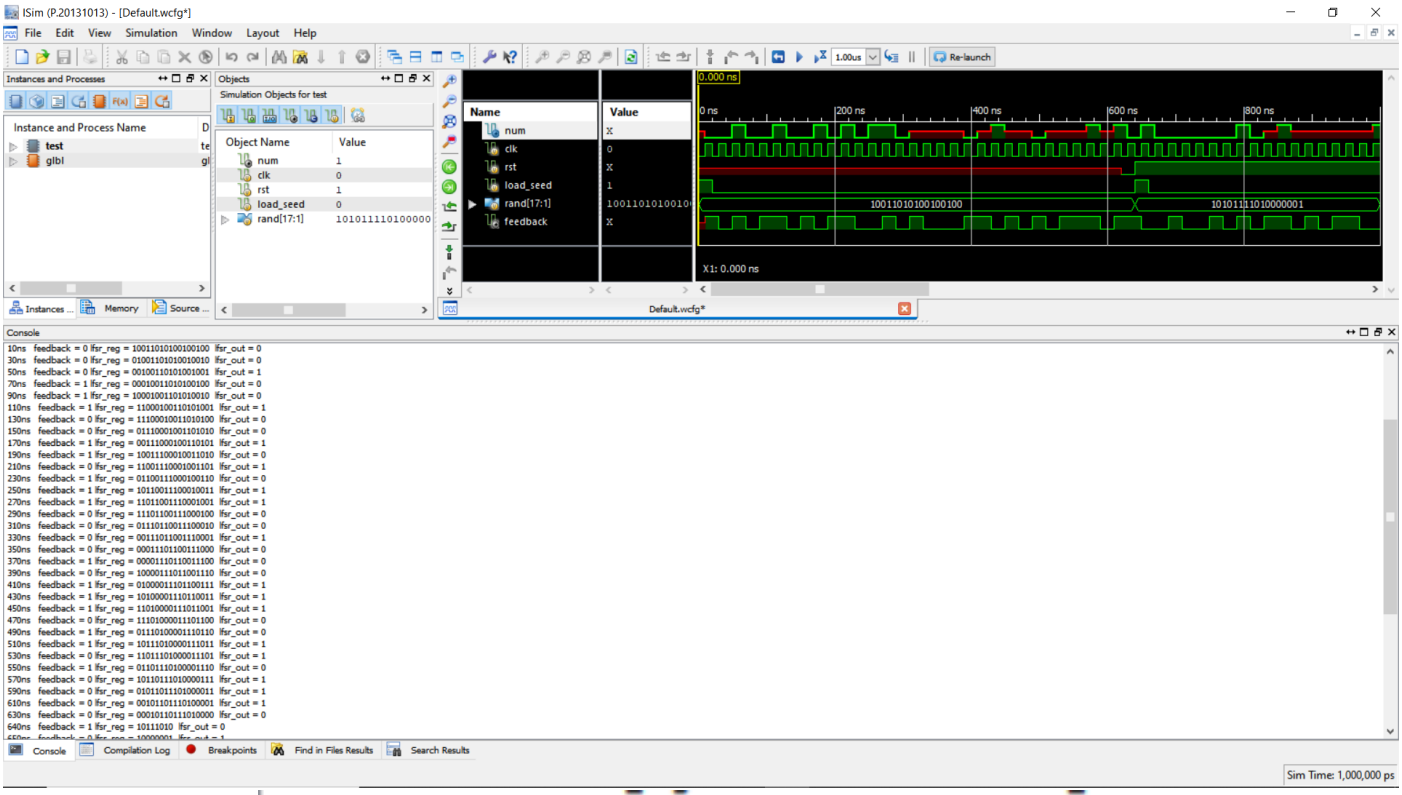
در ابتدا برای 17 بیتی و سپس برای 8 بیتی نمایش داده میشود.

کلاک را هر 10 واحد قرار میدهیم.

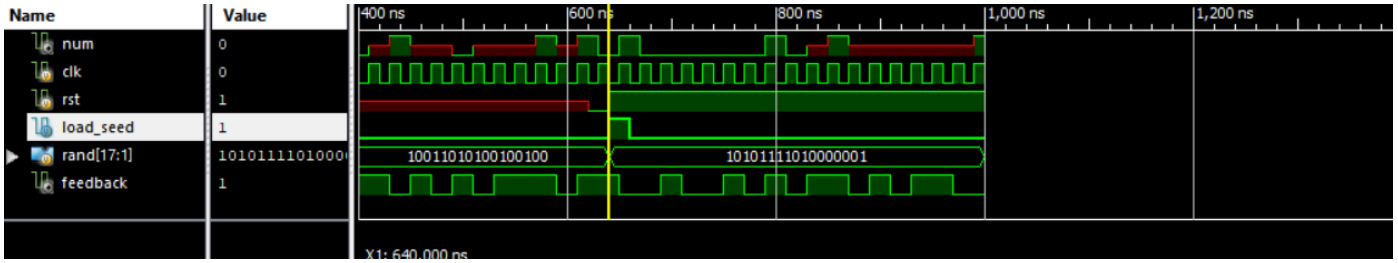
لود در ابتدا 0 است و پس از 15 ثانیه 1 میشود شروع و مقداردهی رندم میکند.

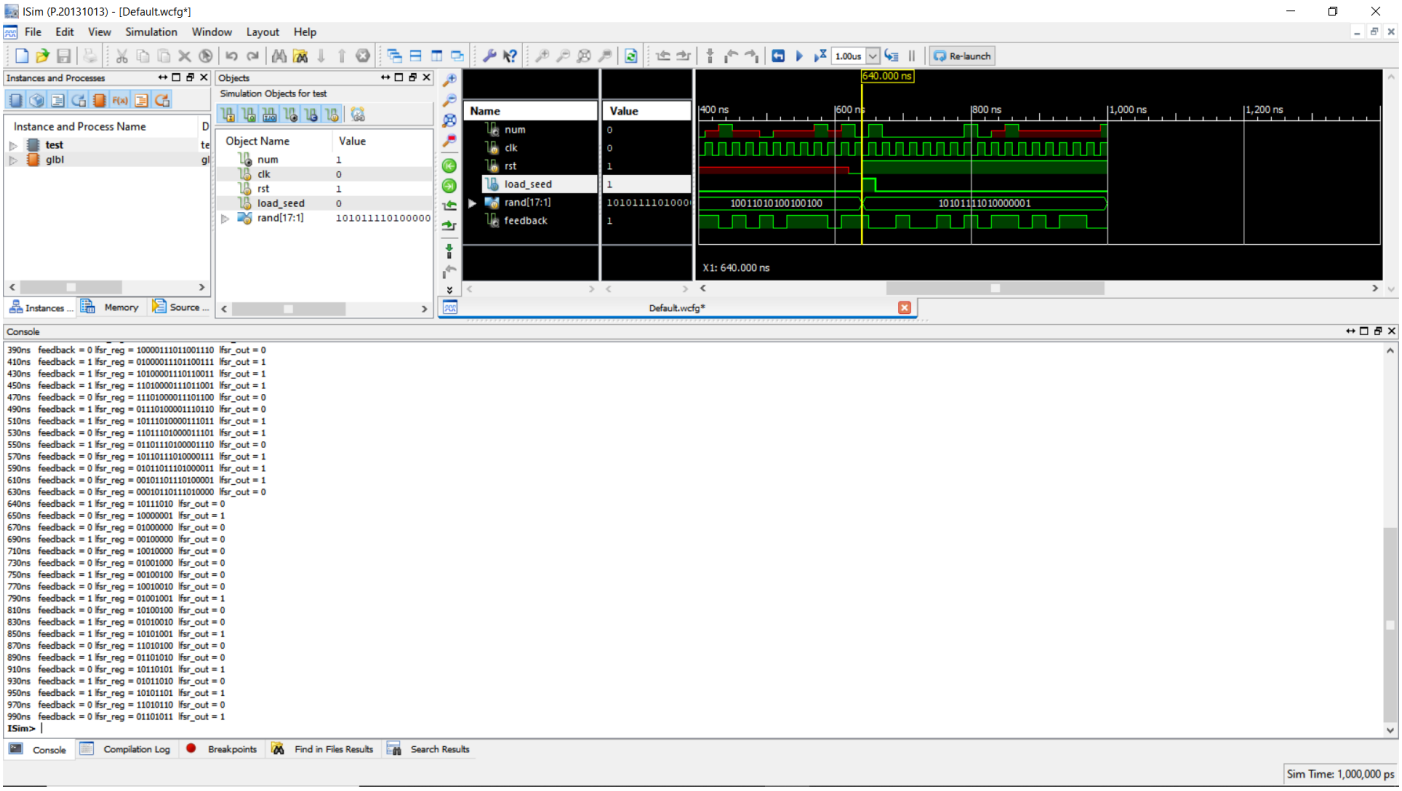



```
10ns   feedback = 0 lfsr_reg = 10011010100100100  lfsr_out = 0
30ns   feedback = 0 lfsr_reg = 01001101010010010  lfsr_out = 0
50ns   feedback = 0 lfsr_reg = 00100110101001001  lfsr_out = 1
70ns   feedback = 1 lfsr_reg = 00010011010100100  lfsr_out = 0
90ns   feedback = 1 lfsr_reg = 10001001101010010  lfsr_out = 0
110ns  feedback = 1 lfsr_reg = 11000100110101001  lfsr_out = 1
130ns  feedback = 0 lfsr_reg = 11100010011010100  lfsr_out = 0
150ns  feedback = 0 lfsr_reg = 01110001001101010  lfsr_out = 0
170ns  feedback = 1 lfsr_reg = 00111000100110101  lfsr_out = 1
190ns  feedback = 1 lfsr_reg = 10011100010011010  lfsr_out = 0
210ns  feedback = 0 lfsr_reg = 11001110001001101  lfsr_out = 1
230ns  feedback = 1 lfsr_reg = 01100111000100110  lfsr_out = 0
250ns  feedback = 1 lfsr_reg = 10110011100010011  lfsr_out = 1
270ns  feedback = 1 lfsr_reg = 11011001110001001  lfsr_out = 1
290ns  feedback = 0 lfsr_reg = 11101100111000100  lfsr_out = 0
310ns  feedback = 0 lfsr_reg = 01110110011100010  lfsr_out = 0
330ns  feedback = 0 lfsr_reg = 00111011001110001  lfsr_out = 1
350ns  feedback = 0 lfsr_reg = 00011101100111000  lfsr_out = 0
370ns  feedback = 1 lfsr_reg = 00001110110011100  lfsr_out = 0
390ns  feedback = 0 lfsr_reg = 10000111011001110  lfsr_out = 0
410ns  feedback = 1 lfsr_reg = 01000011101100111  lfsr_out = 1
430ns  feedback = 1 lfsr_reg = 10100001110110011  lfsr_out = 1
450ns  feedback = 1 lfsr_reg = 11010000111011001  lfsr_out = 1
470ns  feedback = 0 lfsr_reg = 11101000011101100  lfsr_out = 0
490ns  feedback = 1 lfsr_reg = 01110100001110110  lfsr_out = 0
510ns  feedback = 1 lfsr_reg = 10111010000111011  lfsr_out = 1
530ns  feedback = 0 lfsr_reg = 11011101000011101  lfsr_out = 1
550ns  feedback = 1 lfsr_reg = 01101110100001110  lfsr_out = 0
570ns  feedback = 0 lfsr_reg = 10110111010000111  lfsr_out = 1
590ns  feedback = 0 lfsr_reg = 01011011101000011  lfsr_out = 1
610ns  feedback = 0 lfsr_reg = 00101101110100001  lfsr_out = 1
630ns  feedback = 0 lfsr_reg = 00010110111010000  lfsr_out = 0
```



640ns feedback = 1 lfsr_reg = 10111010 lfsr_out = 0
650ns feedback = 0 lfsr_reg = 10000001 lfsr_out = 1
670ns feedback = 0 lfsr_reg = 01000000 lfsr_out = 0
690ns feedback = 1 lfsr_reg = 00100000 lfsr_out = 0
710ns feedback = 0 lfsr_reg = 10010000 lfsr_out = 0
730ns feedback = 0 lfsr_reg = 01001000 lfsr_out = 0
750ns feedback = 1 lfsr_reg = 00100100 lfsr_out = 0
770ns feedback = 0 lfsr_reg = 10010010 lfsr_out = 0
790ns feedback = 1 lfsr_reg = 01001001 lfsr_out = 1
810ns feedback = 0 lfsr_reg = 10100100 lfsr_out = 0
830ns feedback = 1 lfsr_reg = 01010010 lfsr_out = 0
850ns feedback = 1 lfsr_reg = 10101001 lfsr_out = 1
870ns feedback = 0 lfsr_reg = 11010100 lfsr_out = 0
890ns feedback = 1 lfsr_reg = 01101010 lfsr_out = 0
910ns feedback = 0 lfsr_reg = 10110101 lfsr_out = 1
930ns feedback = 1 lfsr_reg = 01011010 lfsr_out = 0
950ns feedback = 1 lfsr_reg = 10101101 lfsr_out = 1
970ns feedback = 0 lfsr_reg = 11010110 lfsr_out = 0
990ns feedback = 0 lfsr_reg = 01101011 lfsr_out = 1


Name
Value
num
0
clk
0
rst
1
load_seed
1
rand[17:1]
feedback
1
400 ns
600 ns
800 ns
1,000 ns
1,200 ns
1001101010010010 0
1010111101000000 1
X1: 640.000 ns


ISim (P.20131013) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
Re-launch
Instances and Processes
Instance and Process Name
test
glbl
Objects
Simulation Objects for test
Object Name
Value
num 1
clk 0
rst 1
load_seed 0
rand[17:1] 1010111101000000
Name
Value
num 0
clk 0
rst 1
load_seed 1
rand[17:1] 1010111101000
feedback 1
640.000 ns
400 ns
600 ns
800 ns
1,000 ns
1,200 ns
1001101010010010 0
1010111101000000 1
X1: 640.000 ns
Instances ...
Memory
Source ...
Default.wcfg*
Console
390ns feedback = 0 lfsr_reg = 10000111011001110 lfsr_out = 0
410ns feedback = 1 lfsr_reg = 01000011101100111 lfsr_out = 1
430ns feedback = 1 lfsr_reg = 10100001110110011 lfsr_out = 1
450ns feedback = 1 lfsr_reg = 11010000111011001 lfsr_out = 1
470ns feedback = 0 lfsr_reg = 11101000011101100 lfsr_out = 0
490ns feedback = 1 lfsr_reg = 01110100001110110 lfsr_out = 0
510ns feedback = 1 lfsr_reg = 10111010000111011 lfsr_out = 1
530ns feedback = 0 lfsr_reg = 11011101000011101 lfsr_out = 1
550ns feedback = 1 lfsr_reg = 01101110100001110 lfsr_out = 0
570ns feedback = 0 lfsr_reg = 10110111010000111 lfsr_out = 1
590ns feedback = 0 lfsr_reg = 01011011101000011 lfsr_out = 1
610ns feedback = 0 lfsr_reg = 00101101110100001 lfsr_out = 1
630ns feedback = 0 lfsr_reg = 00010110111010000 lfsr_out = 0
640ns feedback = 1 lfsr_reg = 10111010 lfsr_out = 0
650ns feedback = 0 lfsr_reg = 10000001 lfsr_out = 1
670ns feedback = 0 lfsr_reg = 01000000 lfsr_out = 0
690ns feedback = 1 lfsr_reg = 00100000 lfsr_out = 0
710ns feedback = 0 lfsr_reg = 10010000 lfsr_out = 0
730ns feedback = 0 lfsr_reg = 01001000 lfsr_out = 0
750ns feedback = 1 lfsr_reg = 00100100 lfsr_out = 0
770ns feedback = 0 lfsr_reg = 10010010 lfsr_out = 0
790ns feedback = 1 lfsr_reg = 01001001 lfsr_out = 1
810ns feedback = 0 lfsr_reg = 10100100 lfsr_out = 0
830ns feedback = 1 lfsr_reg = 01010010 lfsr_out = 0
850ns feedback = 1 lfsr_reg = 10101001 lfsr_out = 1
870ns feedback = 0 lfsr_reg = 11010100 lfsr_out = 0
890ns feedback = 1 lfsr_reg = 01101010 lfsr_out = 0
910ns feedback = 0 lfsr_reg = 10110101 lfsr_out = 1
930ns feedback = 1 lfsr_reg = 01011010 lfsr_out = 0
950ns feedback = 1 lfsr_reg = 10101101 lfsr_out = 1
970ns feedback = 0 lfsr_reg = 11010110 lfsr_out = 0
990ns feedback = 0 lfsr_reg = 01101011 lfsr_out = 1
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps