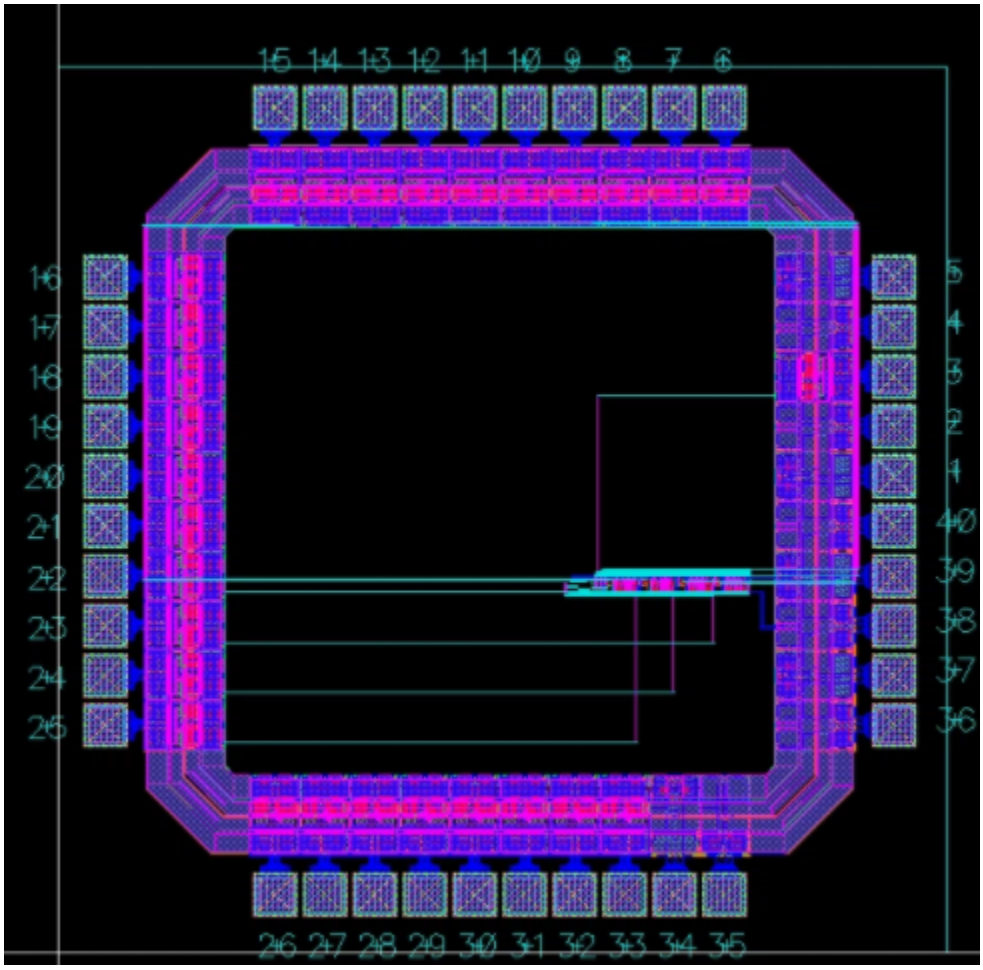
EECS 4612 Group 4:
Pseudo Random Number Generator
LFSR

Designers: Matthew Tolentino 214262729 & David Mounes Flores 215033764



Introduction

Our Pseudo Random Number generator design incorporates four Muxs, four D-FlipFlops, two inverters, and one XOR gate. The Mux has a select bit which is fed into the system to decide between it giving the feed back into the system or the seed value. The seed value is fed into the system as well which determines the pseudo random sequence of numbers generated. Depending on the number of bits that comprises the seed input we will get the length of our sequence which is 2^n-1(n:bits). The four edge triggered flops, function off of a clock input to the system. The System takes the 4 bit seed input and feeds each separate bit into one of the muxes which then will then feed the flops the signal is then cascaded and the last two q bits of flop 3 and 4 go into an XOR gate which then has a feedback loop into flop one.

Inorder to reduce error and increase efficiency we would like to adopt a manufacturing process that uses photolithographic printing to route our chip design, since we would like to make each chip identical chips we can make them in batches on a single silicon wafer. We will then use a binning system to use a wafter sorting process which will allow us to verify if all our chips perform as our design intended.

Contribution Summary

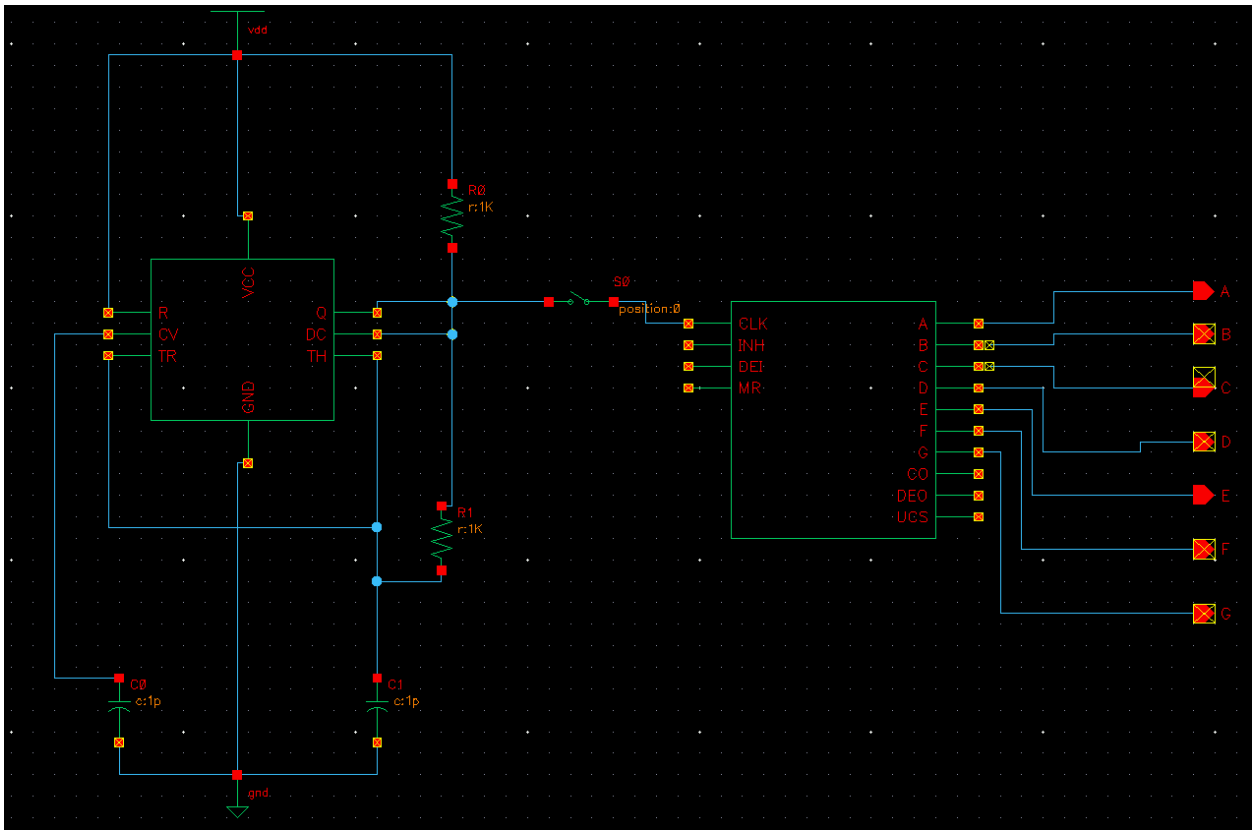
David (Tasks)	Matthew (Tasks)
Verilog code	XOR Schematic
System Schematic	Pad Frame Schematic
Verilog Code / Self checking Test Bench	XOR Layout
Test Fixture	System Layout
Chip Out Routing & GND VDD Routing	Pad Frame Layout
Report	Report

Specifications

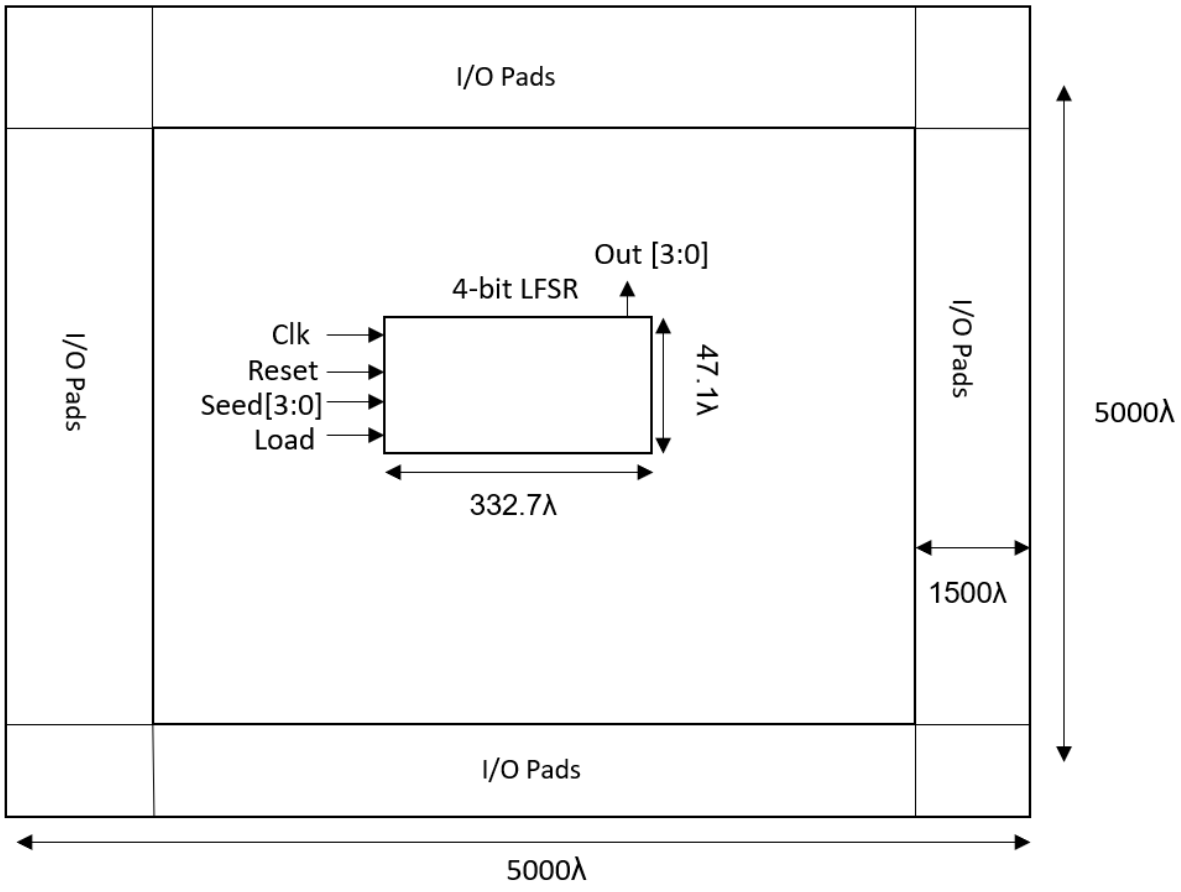
Inputs	Outputs	Direction	Bus Widths	Theory
Clock(clk)		Into system	1bit	The clock input for the synchronous logic
Reset(Rst)		Into system	1bit	Reset to for the FlipFlops
Seed[3:0]	Out[3:0]	Seed:Into system Out:Out of system	4bits	Seed: Is what determines our pseudo random sequence Output: Is the output of our pseudo random sequence
Load		Into system	1bit	The select for muxes to determine to pass feedback or the seed

Floorplan

Initial Floor Plan:



Final Floor Plan:



As seen above our initial floor plan and design needed a complete overhaul which we did initially and we had some analog components and a sorta of random value using a 555 timer into a 7 seg display. Our new design incorporated a 4 bit LFSR to perform the pseudo random sequence of numbers.

Post Fabrication Test Plan

We will hook up our Output I/O port to the chip, and all other necessary input ports (clock, Rst, seed, load). We will then check the binary output from the Output I/O port Out and see if it matches the pseudo random sequence that is expected for the seed input. As previously mentioned since we want our chips to be identical in functionality we will also use wafer sort to figure out if our design constraints are being met.

Design Time

Component	Time
XOR Schematic	1 Hour
System Schematic	4 Hours
Pad Frame Schematic	2 Hour
XOR Layout	3 Hours
System Layout	6 Hours
Pad Frame Layout	2 Hour
Chip Out Routing & GND VDD Routing	1 Hour
Verilog Code / Self checking Test Bench	5 Hours
Test Fixture	3 Hour

File Locations

File Name	File Location
lfsrtst.v	https://github.com/tinozzz/EECS_4612_Project_2_David_and_Matthew/blob/main/lfsr.sv
lfsr.sv	https://github.com/tinozzz/EECS_4612_Project_2_David_and_Matthew/blob/main/lfsr.sv
lfsr.tv	https://github.com/tinozzz/EECS_4612_Project_2_David_and_Matthew/blob/main/lfsr.tv
testfixture.verilog	https://github.com/tinozzz/EECS_4612_Project_2_David_and_Matthew/blob/main/testfixture.verilog
D-flipflop	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/D_flipflop
LFSR_4bit	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/LFSR_4bit
LFSR_4bit_padframe	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/LFSR_4bit_padframe
MUX2X2	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/MUX2X2
flopenr_dp_1x	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/flopenr_dp_1x
nand2	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/nand2
padframe	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/padframe
xor	https://github.com/tinozzz/EECS_4612_Project_2_Group_4/tree/main/xor

Verilog Code and Testbench

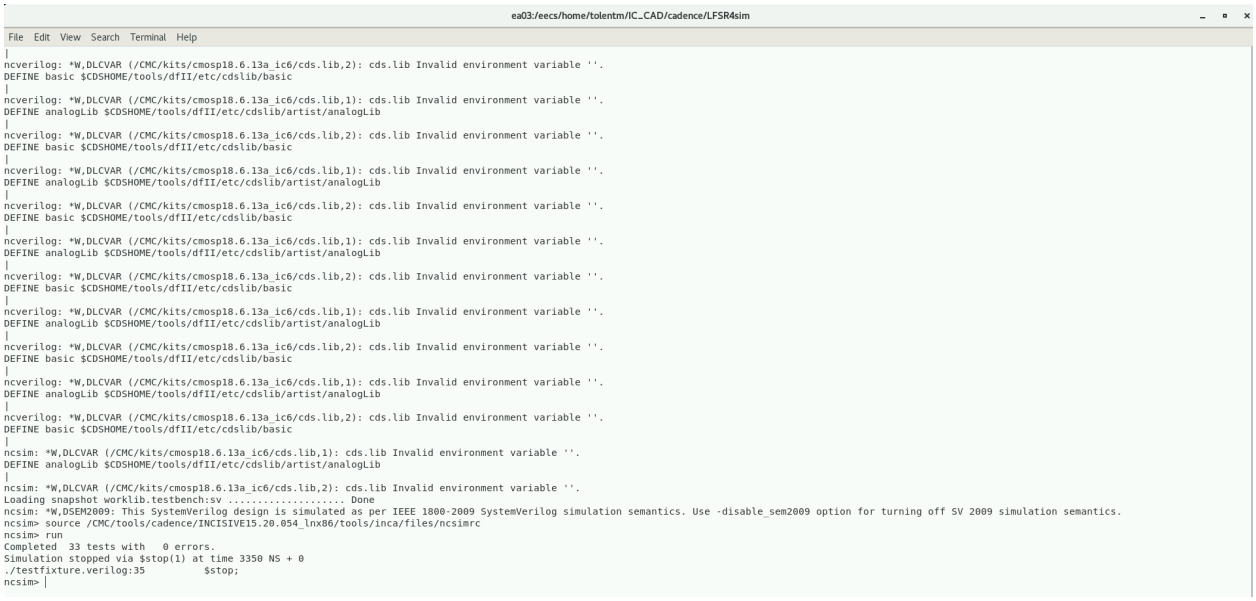
```
module lfsrtst;
reg clk;
reg rst;
reg [3:0] seed;
reg load;
wire [3:0]out;
wire q;
lfsr1 dut(q, clk, rst, seed, load,out);


initial
begin
clk = 0;
load = 0;
seed = 0;
rst = 0;
#10 rst = 1;
#10 rst = 0;
end


always
#50 clk = !clk;

initial begin
#100 seed = 4'b1001;
load = 1;
#100 load = 0;
end

endmodule


module lfsr1(q, clk, rst, seed, load,out);
output q;
input [3:0] seed;
input load;
input rst;
output [3:0]out;
wire [3:0] outS;
wire [3:0] inS;
input clk;
```

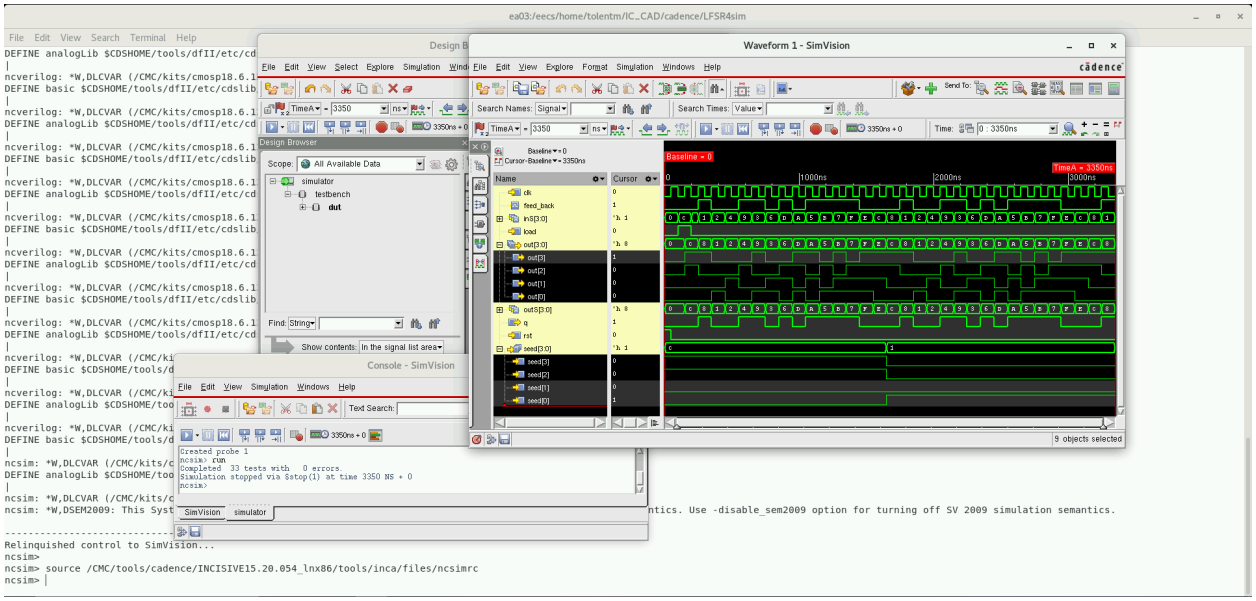
```
flipflop FlipF[3:0] (outS, clk, rst, inS);
mux MUXS[3:0] (inS, load, seed, {outS[2],outS[1],outS[0],feed_back});

xor XORG(feed_back, outS[2], outS[3]);

assign q = feed_back;
assign out = outS;

Endmodule

module mux(q, select, a, b);
output q;
reg q;
input select, a, b;
wire notselect;


always @(select or notselect or a or b)
q = (select && a)||(notselect && b);

not(notselect, select);
      endmodule

module flipflop(q, clk, rst, d);
input clk;
input rst;
input d;
output q;
reg q;
always @(posedge clk or posedge rst)
begin
if (rst)
q = 0;
else
q = d;
end
endmodule
```

Verification

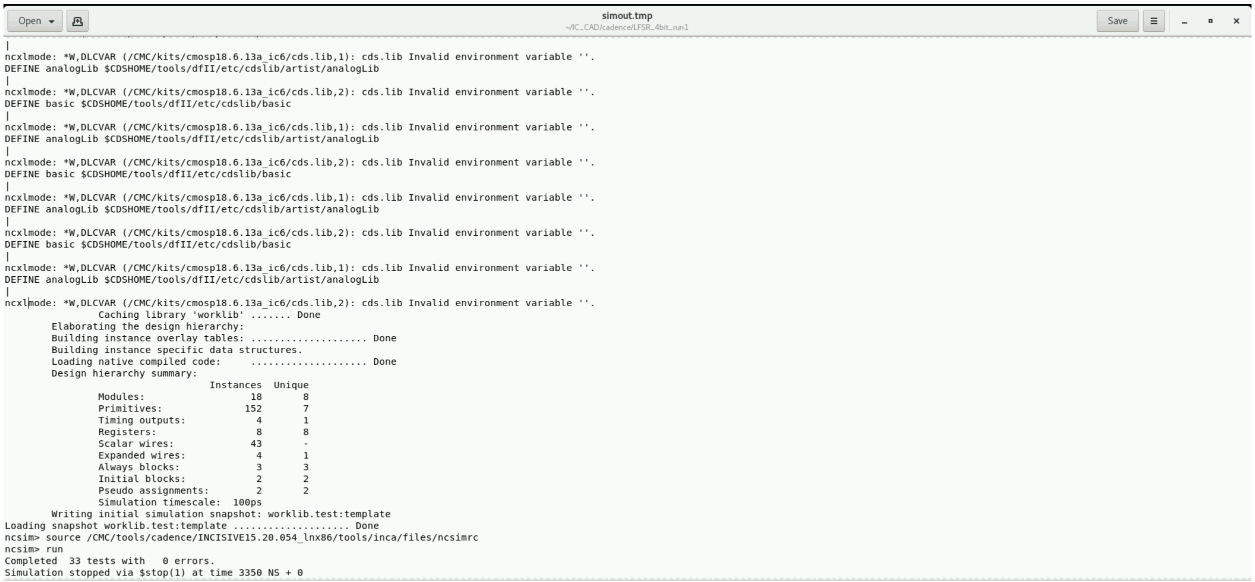
Test	Pass	Fail
Verilog Testbench	Passes	
Schematics Testbench	Passes	
Layout DRC	Passes	
Layout LVS	Passes	

Verilog passes testbench as shown below:

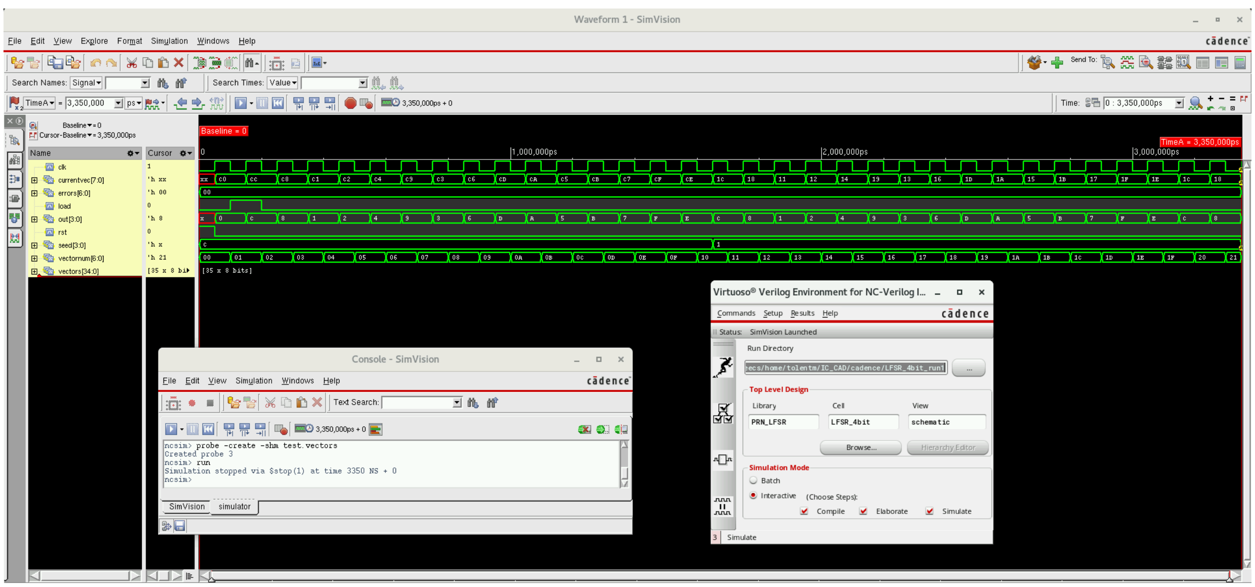




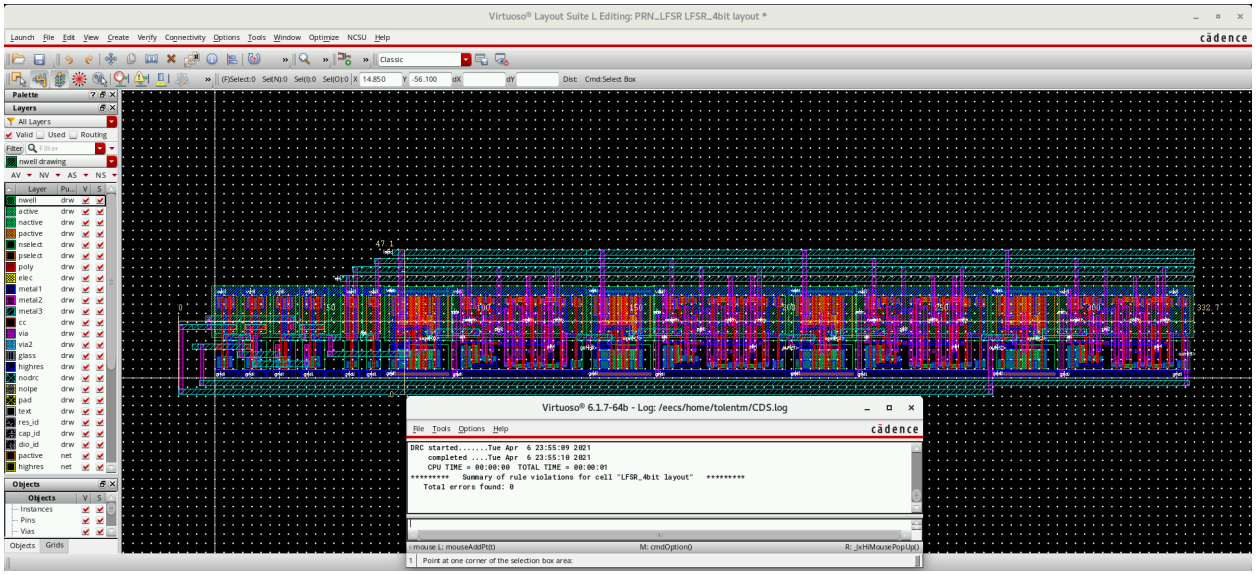
Schematic passes testbench as shown below:



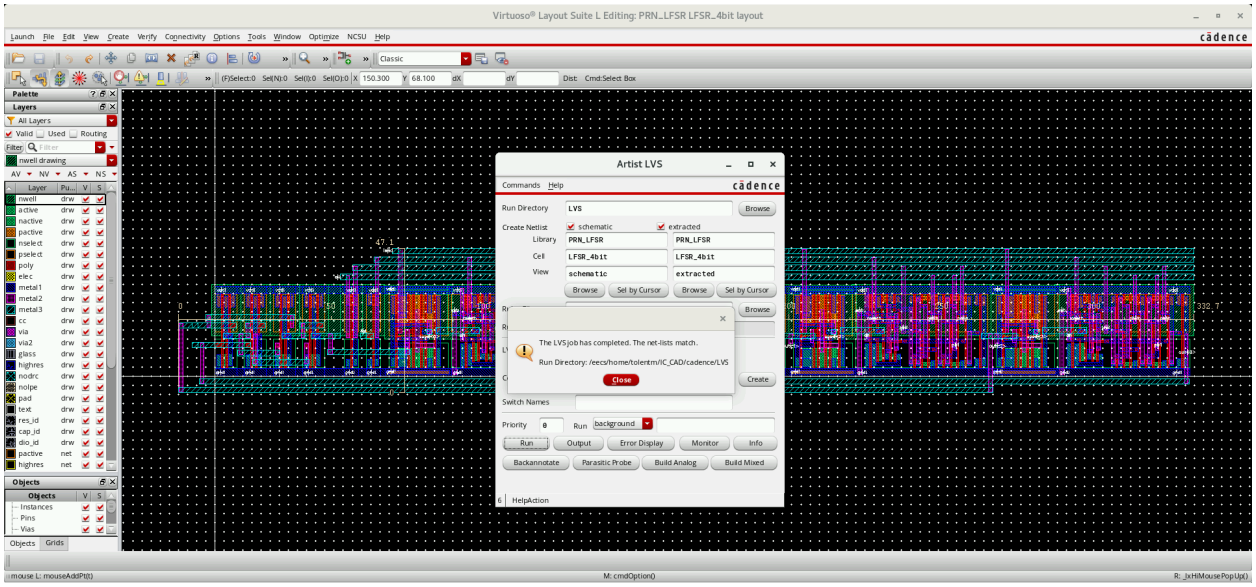
```
simout.tmp
~/IC_CAD/cadence/LFSR_4bit_run1
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,1): cds.lib Invalid environment variable ''.
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,2): cds.lib Invalid environment variable ''.
DEFINE basic $CDSHOME/tools/dfII/etc/cdslib/basic
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,1): cds.lib Invalid environment variable ''.
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,2): cds.lib Invalid environment variable ''.
DEFINE basic $CDSHOME/tools/dfII/etc/cdslib/basic
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,1): cds.lib Invalid environment variable ''.
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,2): cds.lib Invalid environment variable ''.
DEFINE basic $CDSHOME/tools/dfII/etc/cdslib/basic
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,1): cds.lib Invalid environment variable ''.
DEFINE analogLib $CDSHOME/tools/dfII/etc/cdslib/artist/analogLib
|
ncxlmode: *W,DLCVAR (/CMC/kits/cmosp18.6.13a_ic6/cds.lib,2): cds.lib Invalid environment variable ''.
        Caching library 'worklib' ....... Done
    Elaborating the design hierarchy:
    Building instance overlay tables: .................... Done
    Building instance specific data structures.
    Loading native compiled code:     .................... Done
    Design hierarchy summary:
                         Instances  Unique
        Modules:                18       8
        Primitives:            152       7
        Timing outputs:          4       1
        Registers:               8       8
        Scalar wires:           43       -
        Expanded wires:          4       1
        Always blocks:           3       3
        Initial blocks:          2       2
        Pseudo assignments:      2       2
        Simulation timescale:  100ps
    Writing initial simulation snapshot: worklib.test:template
Loading snapshot worklib.test:template .................... Done
ncsim> source /CMC/tools/cadence/INCISIVE15.20.054_lnx86/tools/inca/files/ncsimrc
ncsim> run
Completed  33 tests with   0 errors.
Simulation stopped via $stop(1) at time 3350 NS + 0
```



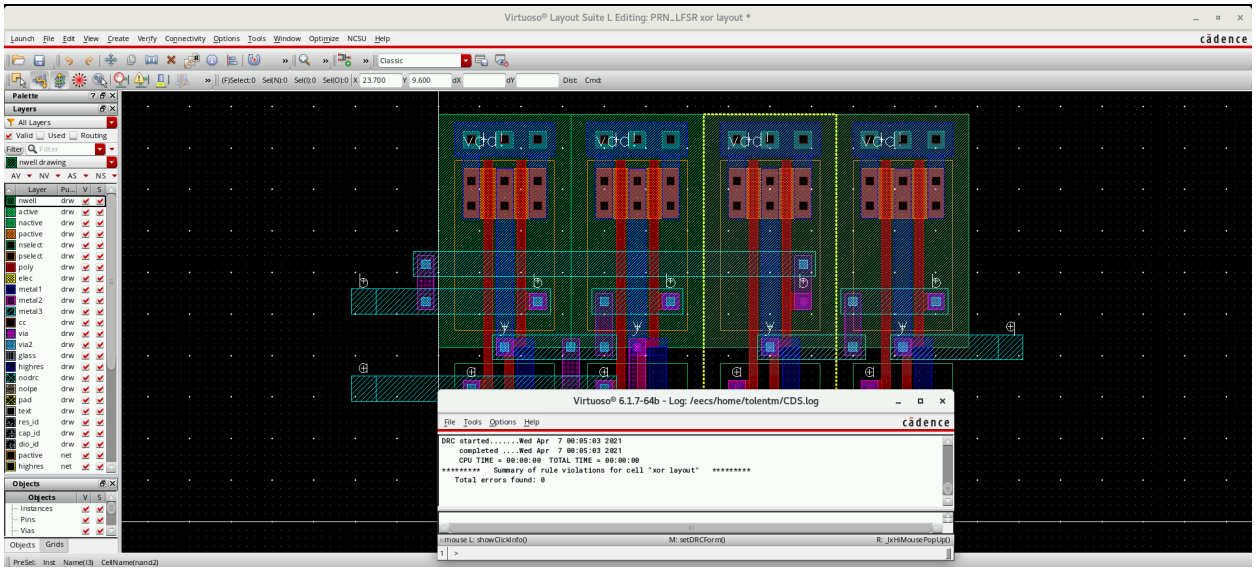
LFSR 4-bit layout passes DRC as shown below:



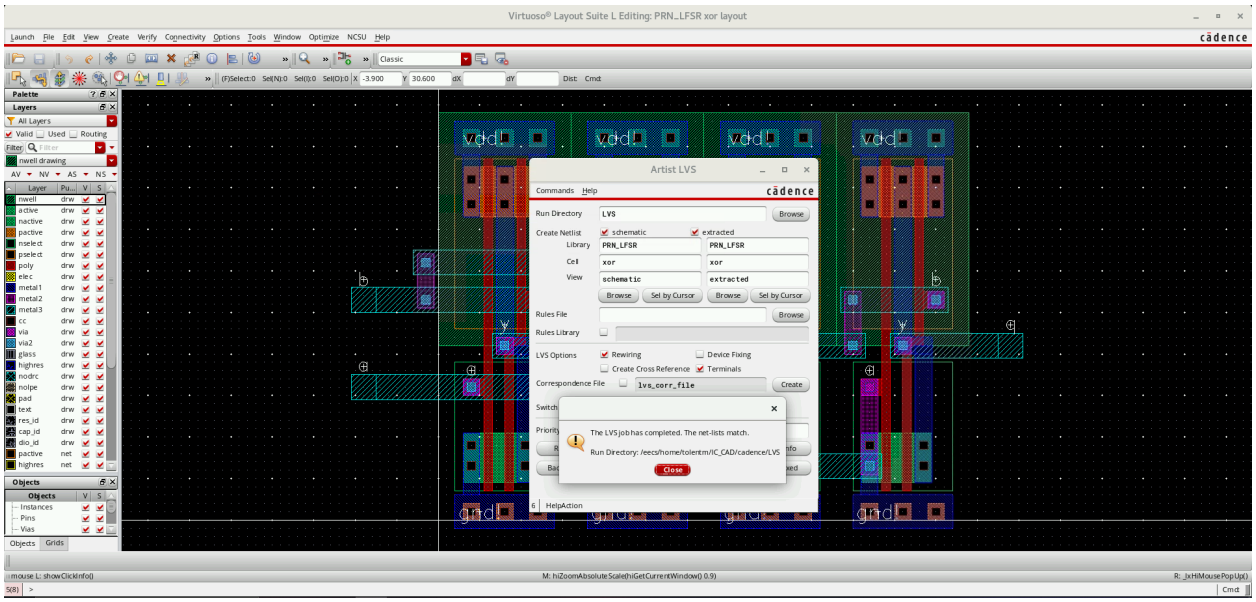
LFSR 4-bit layout passes LVS as shown below:



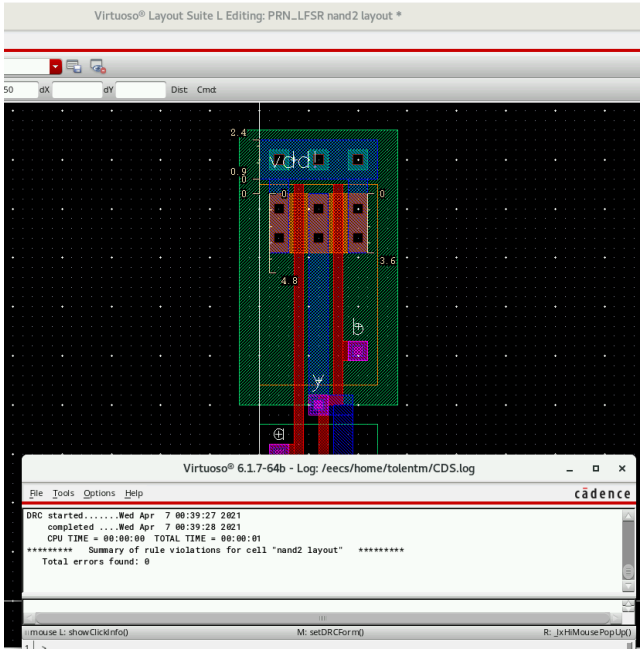
XOR layout passes DRC as shown below:



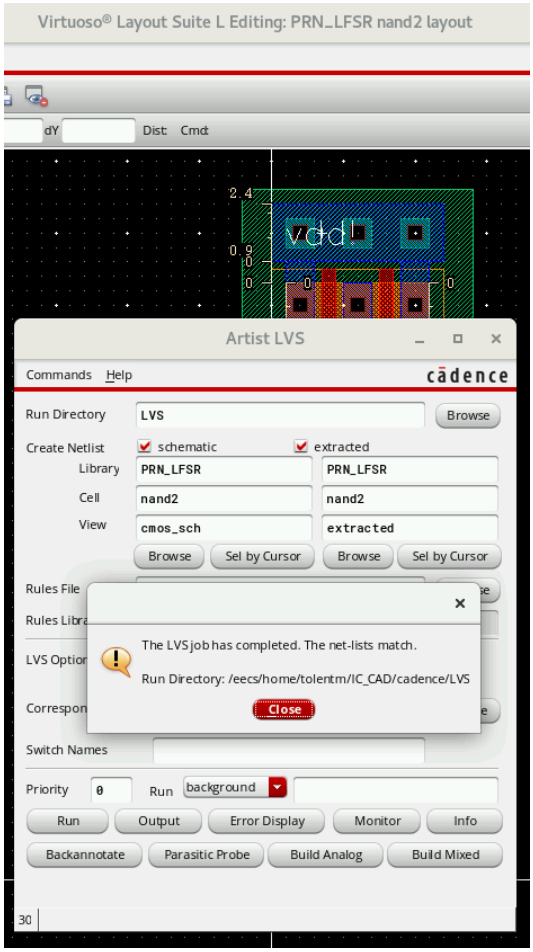
XOR layout passes LVS as shown below:



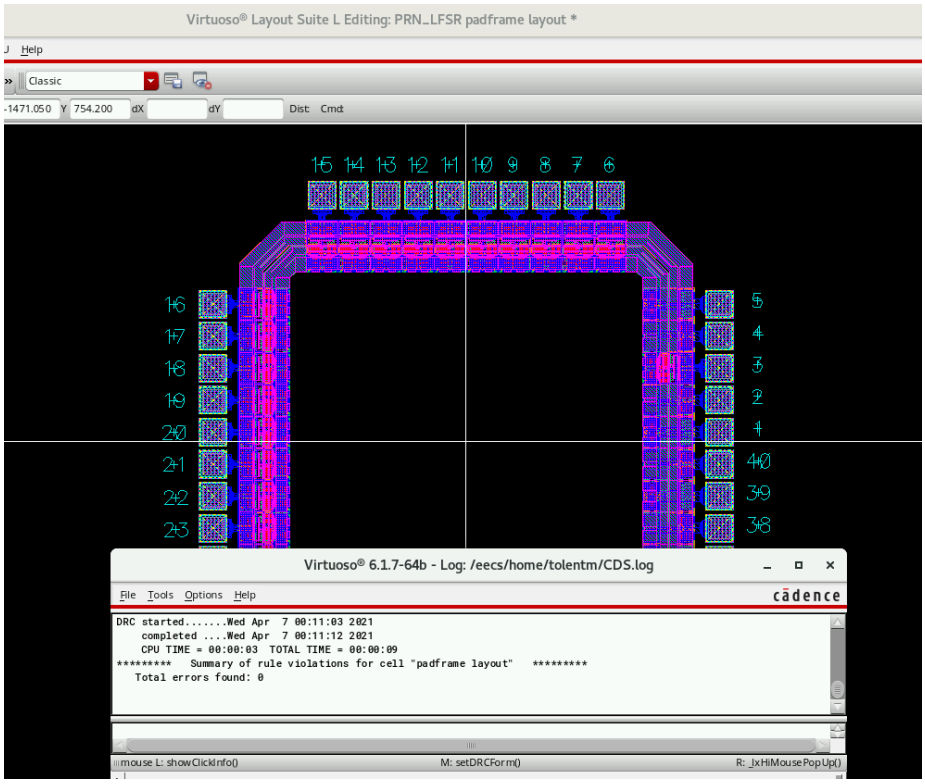
NAND2 layout passes DRC as shown below:



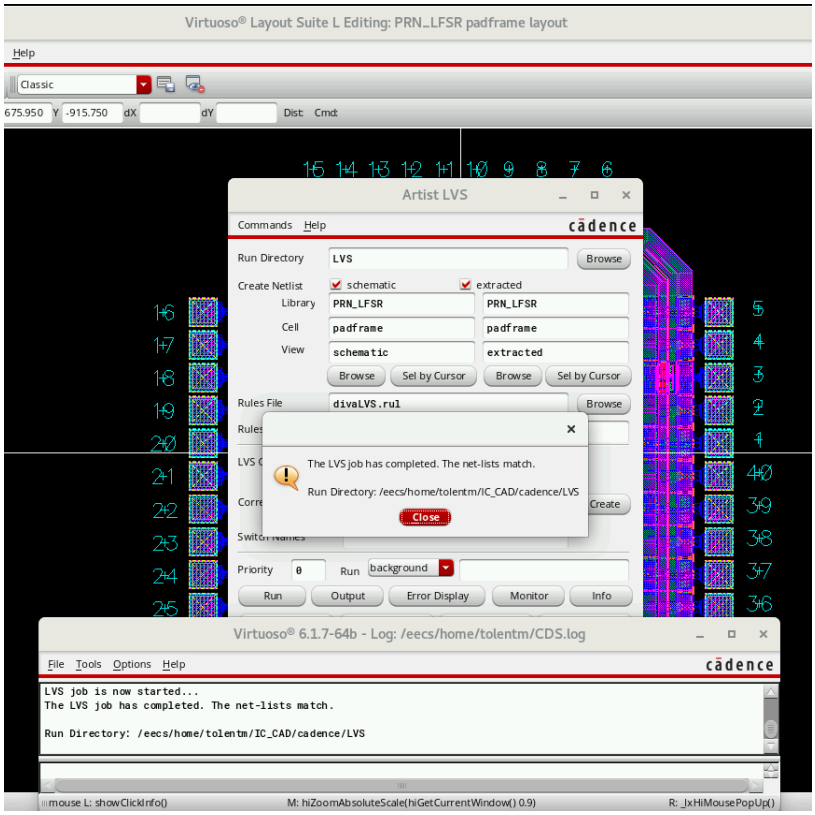
NAND2 layout passes LVS as shown below:



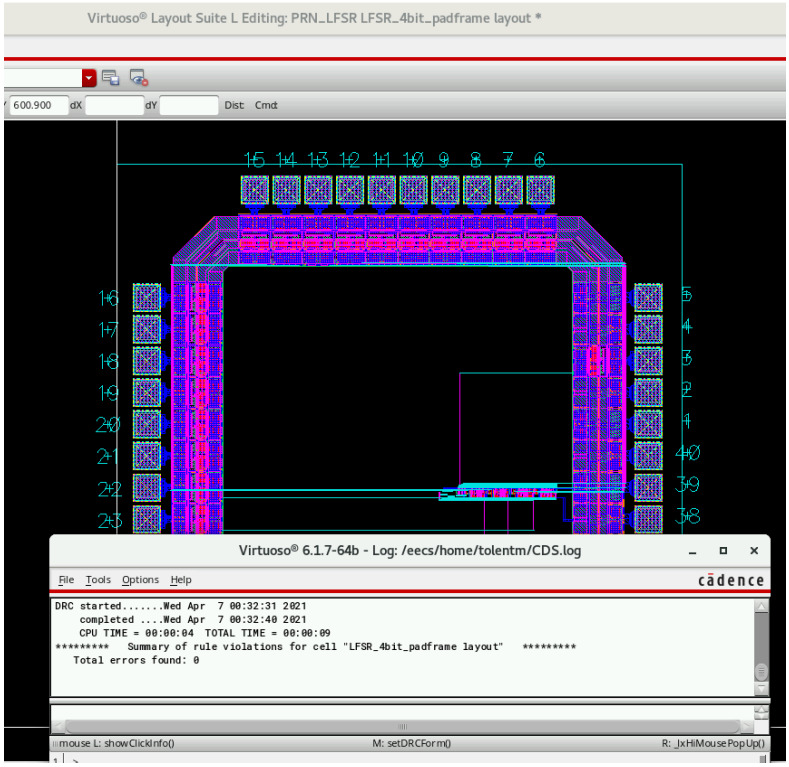
Padframe passes DRC as shown below:



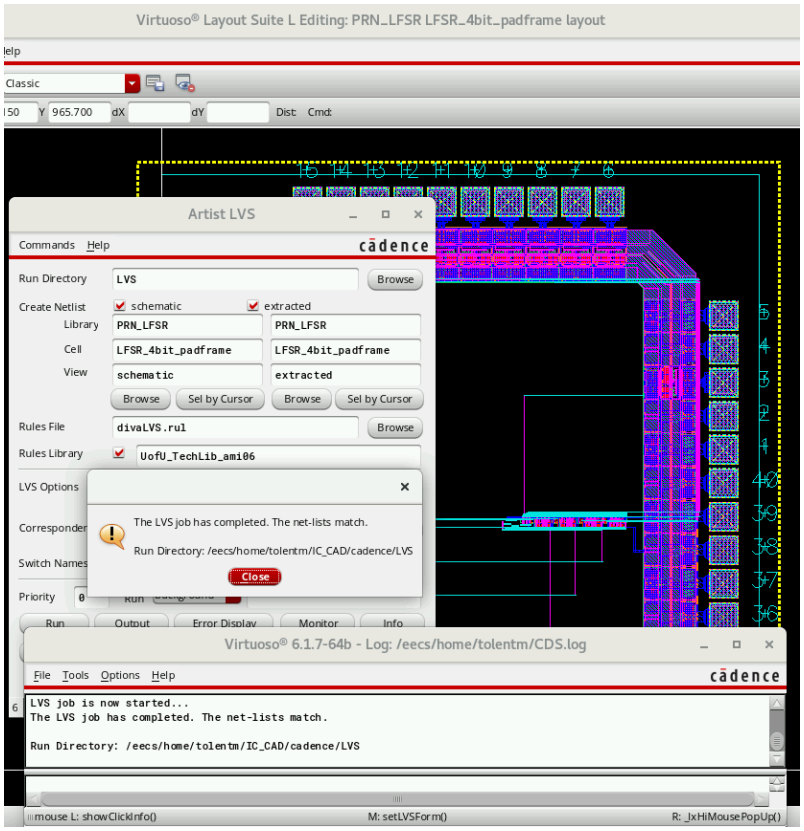
Padframe passes LVS as shown below:



LFSR 4-bit chip passes DRC as shown below:

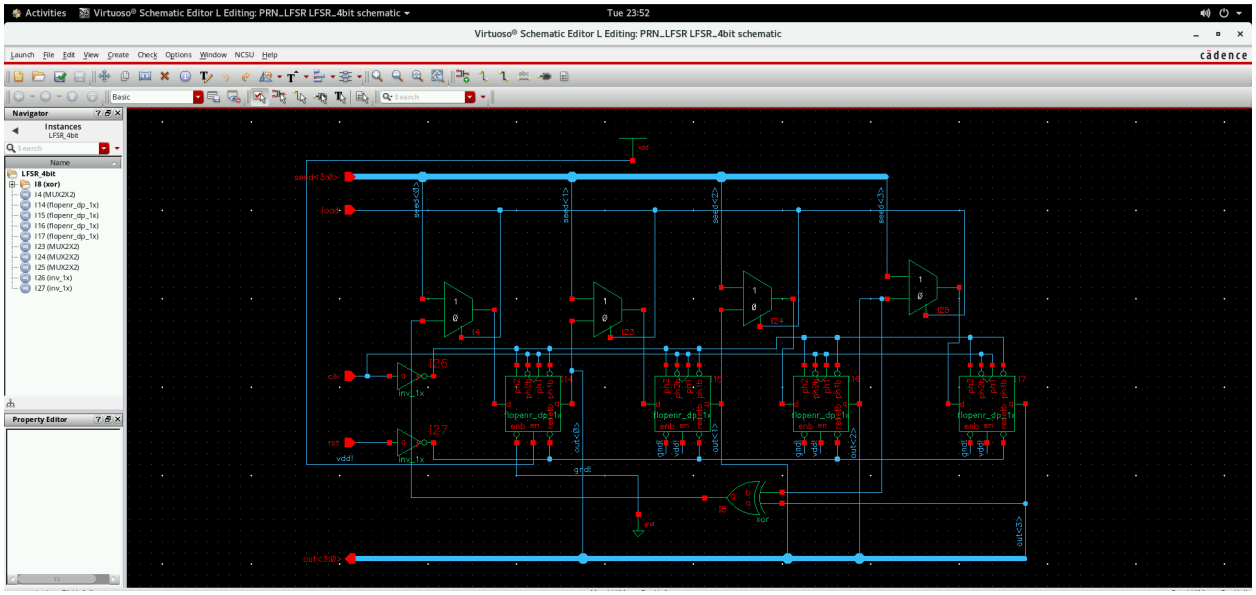


LFSR 4-bit chip passes LVS as shown below:

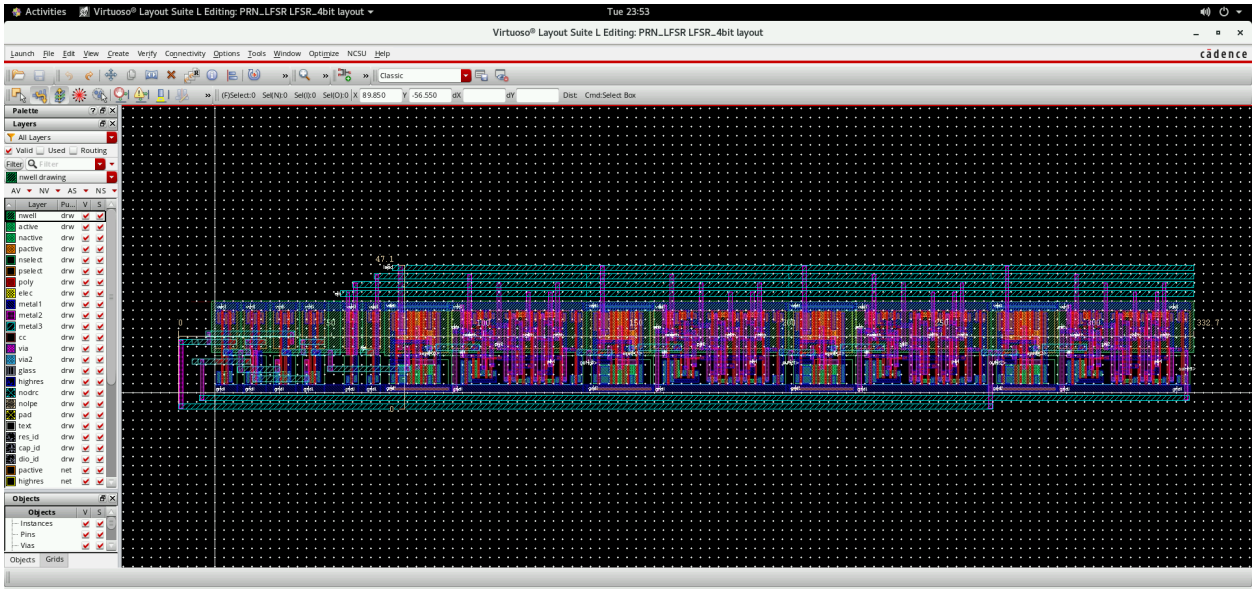


Layouts

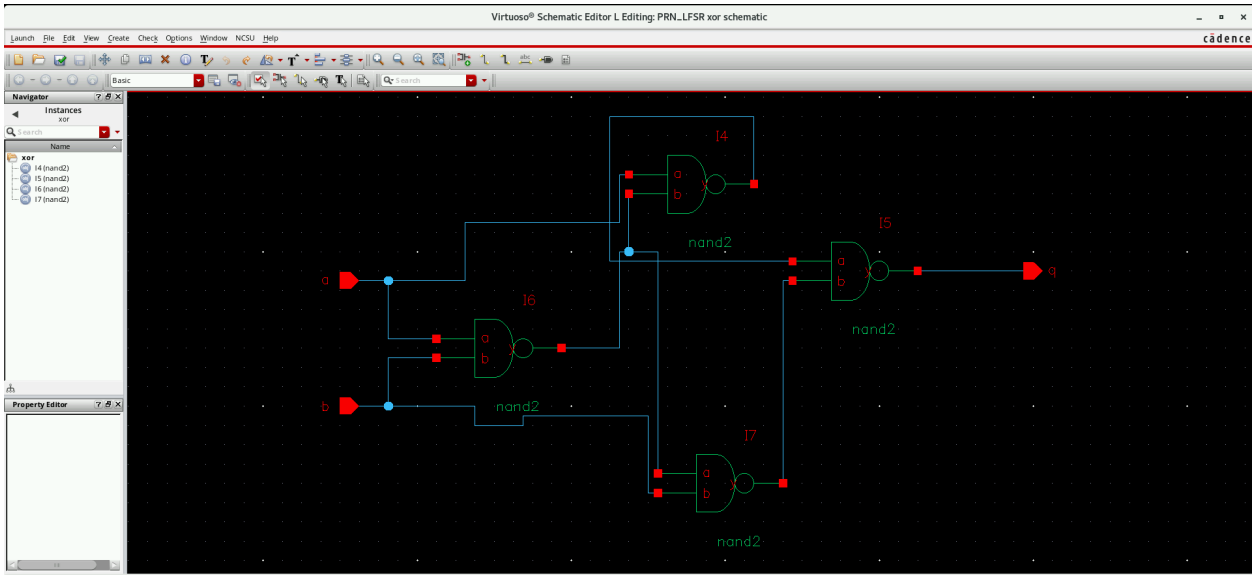
4-bit LFSR Schematic



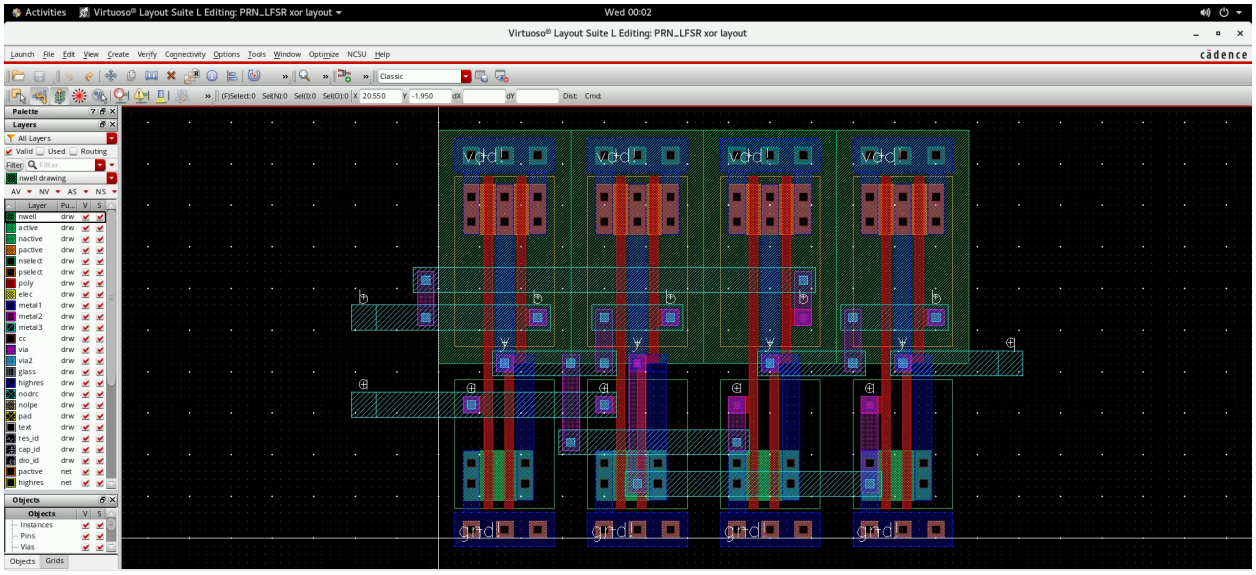
4-bit LFSR Layout



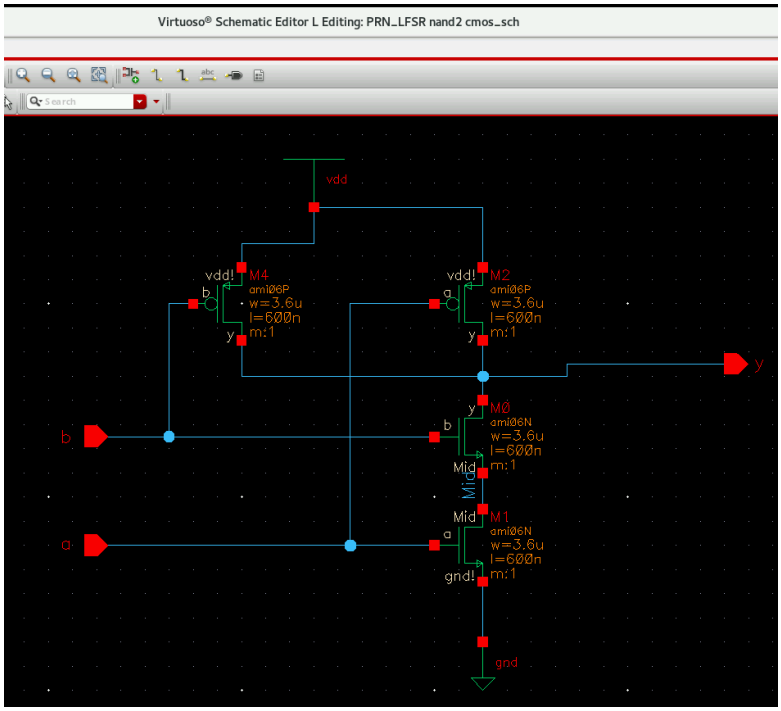
XOR Schematic



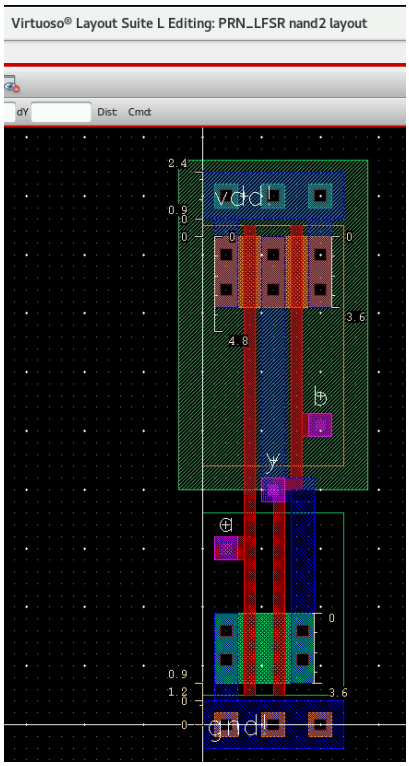
XOR Layout



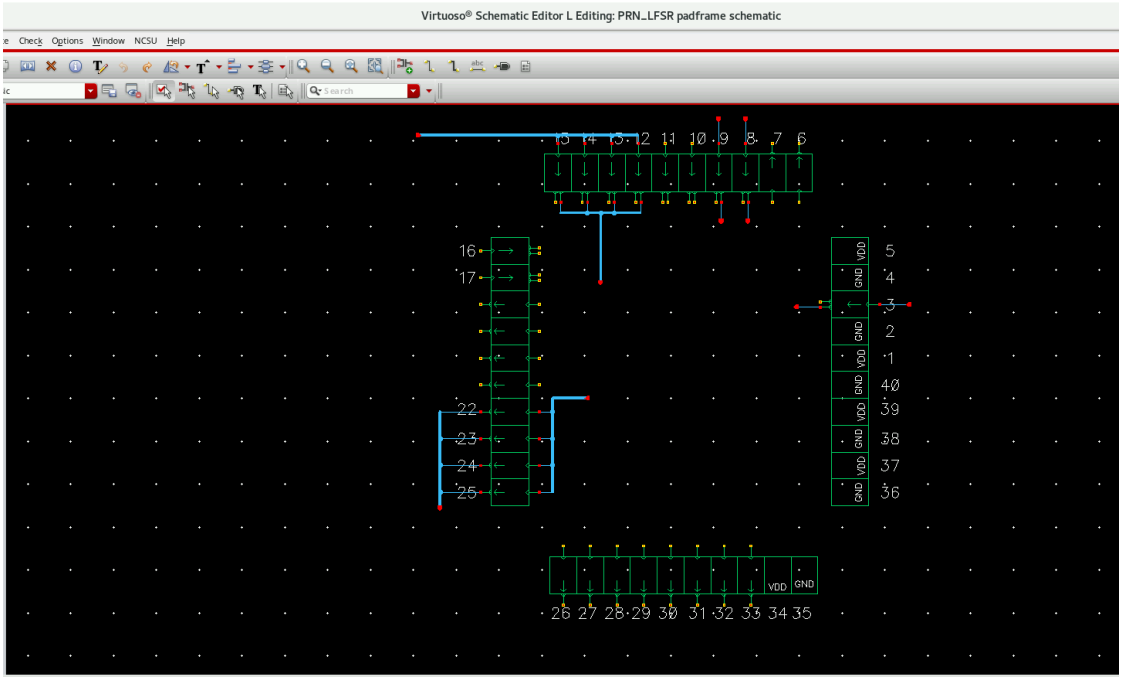
NAND2 Schematic



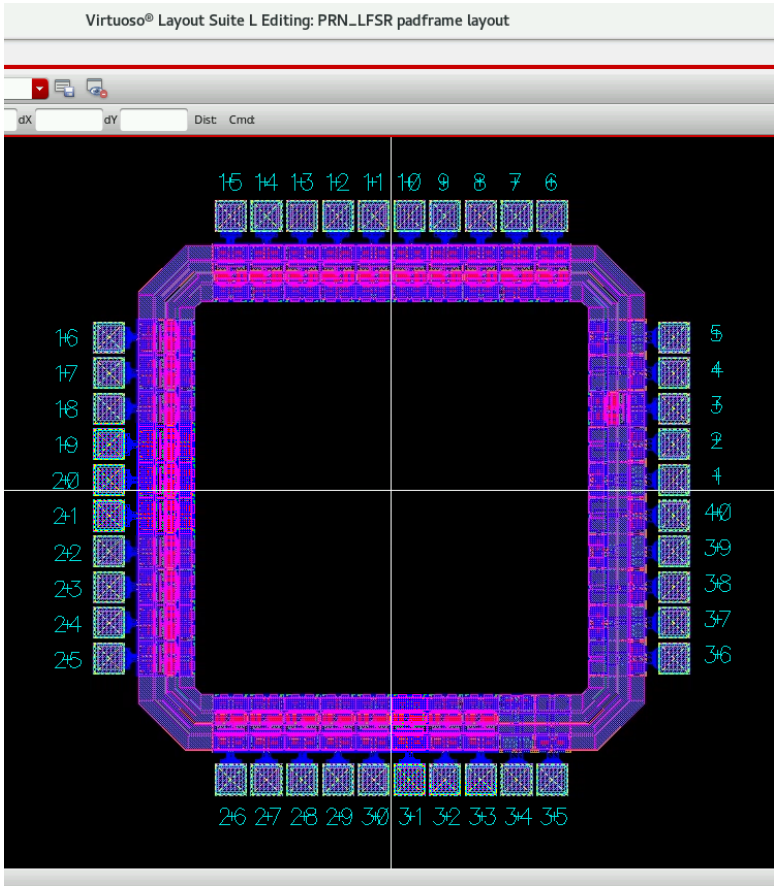
NAND2 Layout



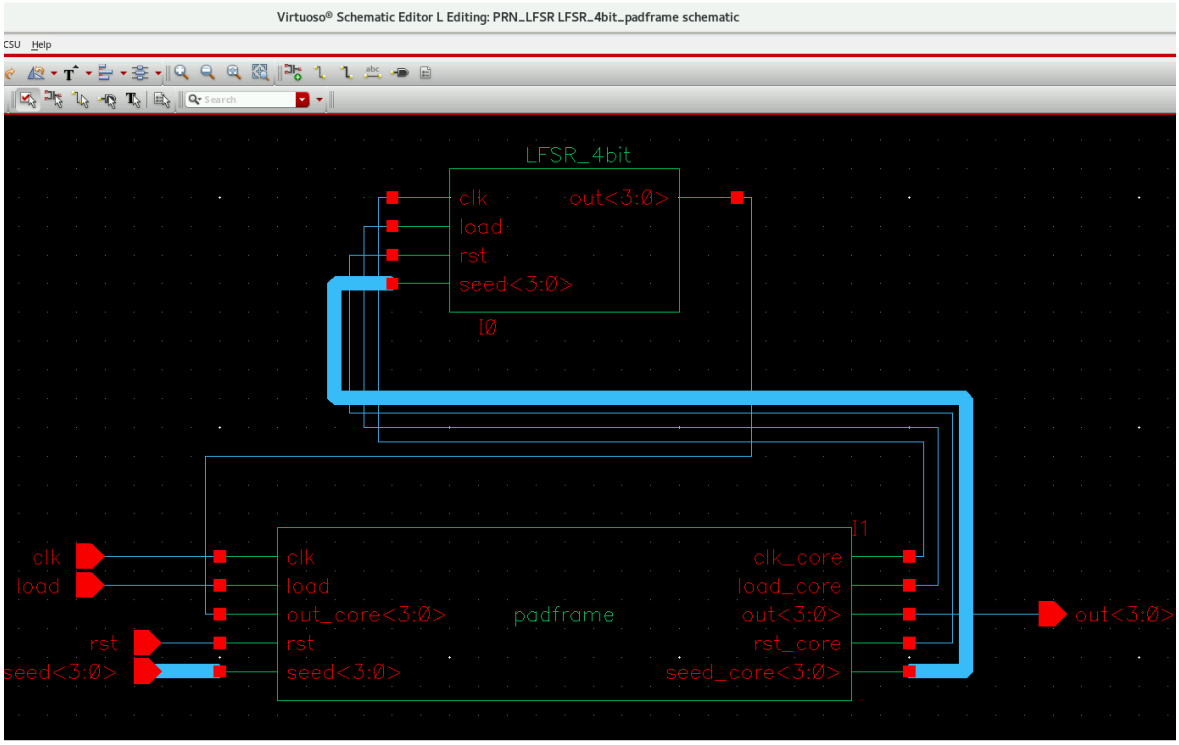
Padframe Schematic



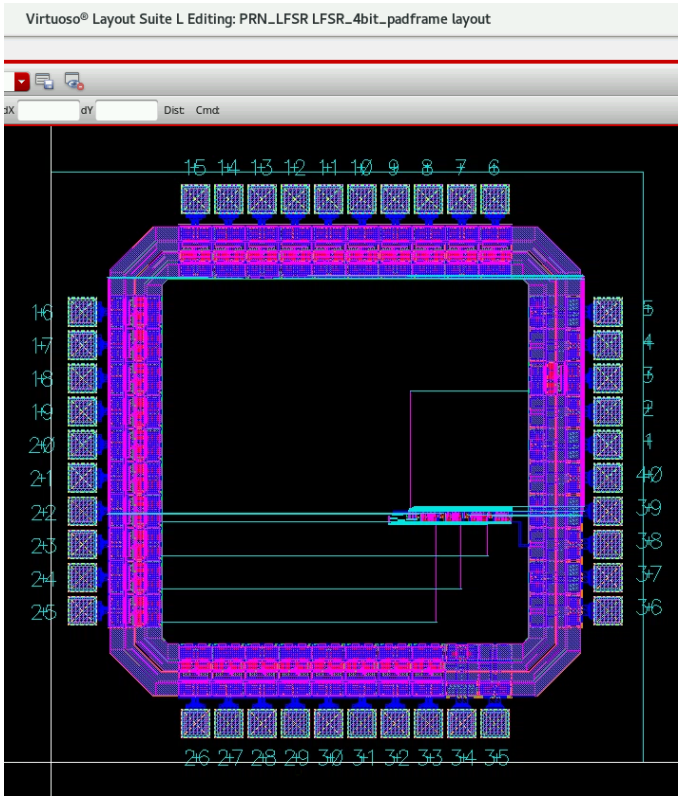
Padframe Layout



LFSR 4-bit Chip Schematic



LFSR 4-bit Chip Layout



References

- Mux, Schematic and layout from UofU_Digital_v1_2
- Inverter, Schematic and layout from muddlib10
- FlipFlop, Schematic and layout from muddlib10