Teaching Labs on Pseudorandom Number Generation

Elizabeth Patitsas

University of Toronto & University of British Columbia

July 5, 2012

Introduction

We teach Pseudorandom Number Generation (PRNG) in our digital logic courses at UBC and Toronto.

In this talk I hope to convince you (and your colleagues) to do the same.

Motivation

Sequential circuitry is often a difficult subject to teach; it's a new paradigm for the students.

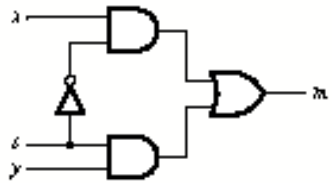
Lab-by-lab surveys identified our lab on sequential circuitry as a weak spot of our digital logic course UBC.

We used "toy problems" for teaching sequential circuitry.

To engage students we introduced a lab on PRNG, as an example of a real-world application of sequential circuitry.

Sequential circuitry vs. combinational circuitry

In combinational circuitry you can represent everything as a truth table, based on possible inputs

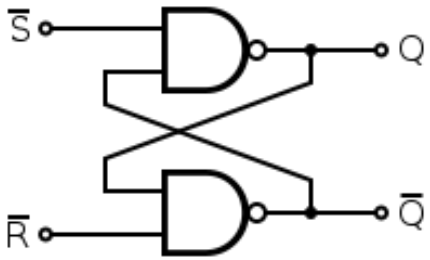


a) Circuit

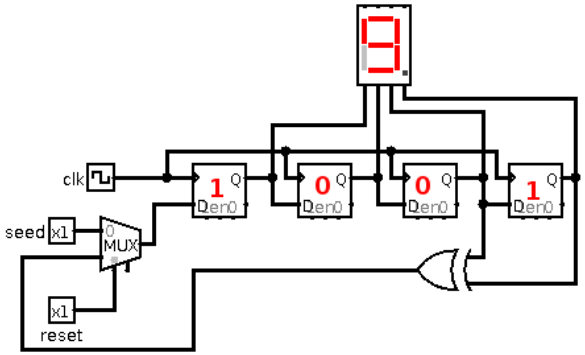
0	
1	

b) Truth table

But in sequential circuitry you use output as input – hence the circuit has a *state*



Linear Feedback Shift Registers (LFSRs)



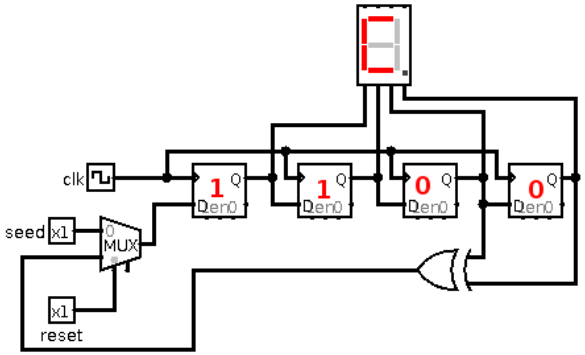
example output:

1	0	0	1	**9**

LFSRs produce pseudorandom numbers by making only a small tweak to a simple shift register.

While far from the best source of pseudorandomness, they're easy to implement in hardware (and are used in the real world!)

Linear Feedback Shift Registers (LFSRs)



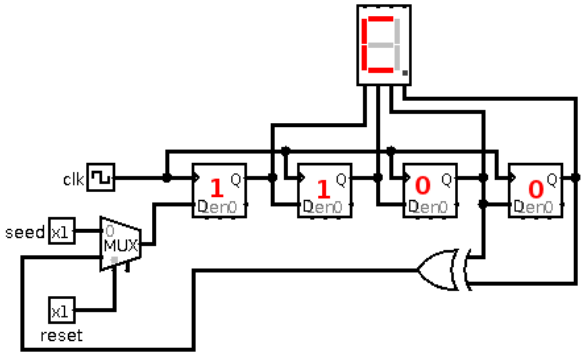
example output:

1	0	0	1	**9**
	1	0	0	

LFSRs produce pseudorandom numbers by making only a small tweak to a simple shift register.

While far from the best source of pseudorandomness, they're easy to implement in hardware (and are used in the real world!)

Linear Feedback Shift Registers (LFSRs)



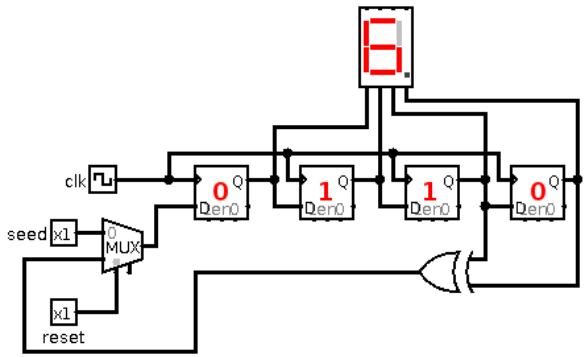
example output:

1	0	0	1	**9**
1	1	0	0	**C**

LFSRs produce pseudorandom numbers by making only a small tweak to a simple shift register.

While far from the best source of pseudorandomness, they're easy to implement in hardware (and are used in the real world!)

Linear Feedback Shift Registers (LFSRs)



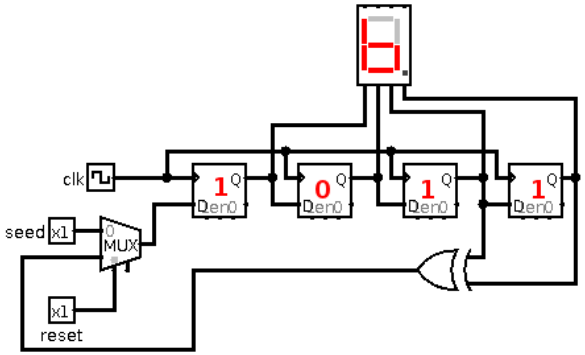
example output:

1	0	0	1	**9**
1	1	0	0	**C**
0	1	1	0	**6**

LFSRs produce pseudorandom numbers by making only a small tweak to a simple shift register.

While far from the best source of pseudorandomness, they're easy to implement in hardware (and are used in the real world!)

Linear Feedback Shift Registers (LFSRs)



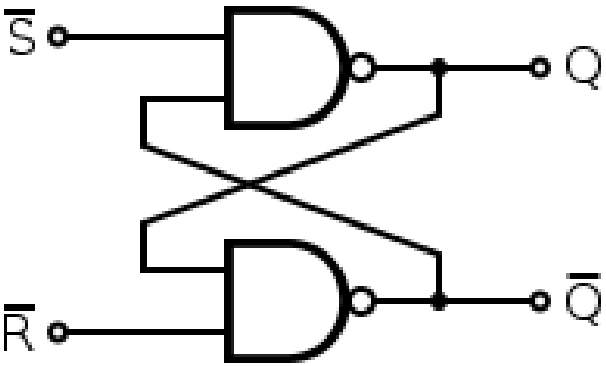
example output:

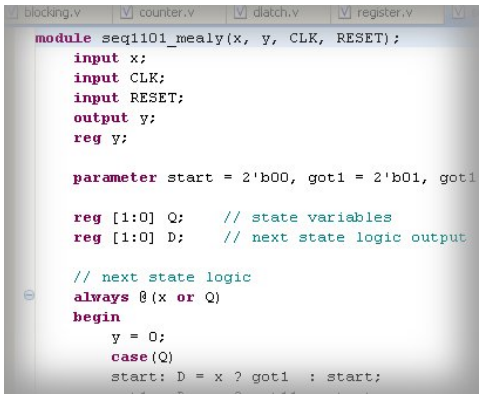
1	0	0	1	**9**
1	1	0	0	**C**
0	1	1	0	**6**
1	0	1	1	**B**

LFSRs produce pseudorandom numbers by making only a small tweak to a simple shift register.

While far from the best source of pseudorandomness, they're easy to implement in hardware (and are used in the real world!)

This is medium-independent



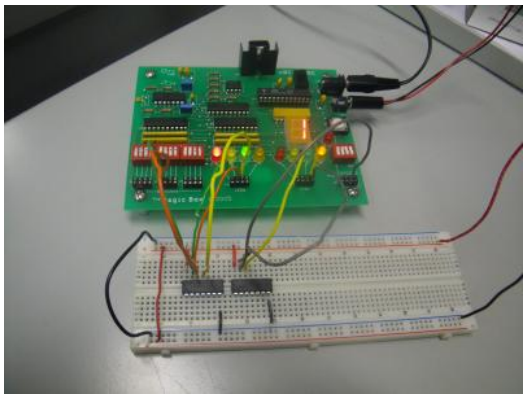


```
blocking.v | counter.v | dlatch.v | register.v
module seq1101_mealy(x, y, CLK, RESET);
    input x;
    input CLK;
    input RESET;
    output y;
    reg y;

    parameter start = 2'b00, got1 = 2'b01, got1

    reg [1:0] Q;    // state variables
    reg [1:0] D;    // next state logic output

    // next state logic
    always @(x or Q)
    begin
        y = 0;
        case(Q)
        start: D = x ? got1  : start;
```



At UBC we've taught this with circuit simulations and breadboarding.

At Toronto we've done it in Verilog.

The UBC Activity

1. Before the lab, students research PRNG, and provide a circuit design for a 4-bit PRNG circuit. (Most do LFSRs, but some do LCG.)
2. In lab, students implement their design and test it (either in a circuit simulator or on a breadboard).
3. They scale it up to 5 bits.
4. We also do an activity with random numbers in C.
5. We finish with open-ended discussions on security and cryptography.

The Toronto Activity

1. We provide students a circuit diagram of a 4-bit LFSR.
2. Before lab, students write Verilog code to implement it, displaying the result to a 7-segment display (a product of a previous lab.)
3. In lab, students download their code to an Altera DE2 board.
4. Students then scale their code up to a 16-bit LFSR.
5. At the end of the term, students do a project – many used their LFSR code to provide random numbers.

Benefits of the Exercise

Sequential circuitry went from being one of the least popular labs to one of the most popular.

It provides a connection between hardware and software; students in the past complained of irrelevance.

It allows us to engage students in a discussion of randomness vs. pseudorandomness.

TAs can discuss what random numbers are used for in CS and in their own research.

More Benefits of the Exercise

Learning PRNG gives the students experience with the concept of seeding.

TAs in courses downstream have reported that students who complete this lab activity now properly seed their C++ code when dealing with stochastic algorithms.

The discussion of seeding bridges into a discussion of security and cryptography (one-time pads and stream ciphers).

Summary

We have successfully taught labs on PRNG at two universities, using three different media

It ties hardware to software that students are learning, and helps in courses downstream.

It's a convenient example for teaching sequential circuitry, and easily added to your curriculum!

Try it Yourself!

The UBC lab is available at `http://www.ugrad.cs.ubc.ca/~cs121/2011W2/Homepage/labs.html`

Acknowledgements:

Lab development: Meghan Allen, Patrice Belleville, Steve Engels, Simon Hastings, Rachel Jordan, Vanessa Kroeker, Kimberly Voll, Steve Wolfman, Bob Woodham, and the TAs of the two courses.

Travel funding: Steve Easterbrook / NSERC

Presentation feedback: Alan, Andrew, Diane, François, Jen, Karen, Marge, Michelle, Steve and Velian.