Lab 13 Report

Kiran Nadkarni

Elizabeth

11/22/19

PRBS Generator

Objectives:

The objective of this lab is to create a PRBS circuit, that receives a message of bits and is able to encrypt those bits to be something random. We then want to decrypt them to read our message out. We do this using knowledge from our previous modules on both PRBS generators and what they are conventionally made from, shift registers. We will be using multiple sets of shift registers, as well as XOR gates to help scramble the bits, as mentioned in the module.

Introduction:

PRBS stands for Pseudo Random Binary Sequence and is used often in cypher transmissions between different parties. The main part of our circuit will be our shift registers, which will move the bits, but we will include XOR gates in between some of these sets of registers to scramble the bits into garbage and also unscramble them to make them readable. From what we have learned, the general skeleton of a PRBS system will have the PRBS generator and the message bits that we want being fed into the previously mentioned XOR gate. This XOR gate then turns our message into garbage that we cannot read. If we want to be able to interpret the bits, we then can feed this garbage and the PRBS generator into another XOR gate, which will then re-generate our original message bits. In general, PRBS circuits abide by the rule of creating a sequence up to 2^n – 1 bits long, where n is the length of the registers.

Procedure:

As mentioned before the procedure for this lab is fairly straightforward once the concepts behind a PRBS is understood.

1- Make a 4-bit register through the of D-flip flops for ease of creation. These will share a clock and reset switch too. The set switch will be different for each flip flop, and the input would be as a normal register is. The output of the last flip flop in this register will then go into the input of a XOR as previously mentioned.
2- This XOR gate has a second input, that we will attach to a second shift register, this one helping to turn our code into garbage. The clock, set, and resets for these flip flops are to be set up the same way as the previous registers were. We will also have another XOR gate with one input attached between the 2nd and 3rd flip flop of this register and the other input attached to the rightmost flip flop on this register, the same as the second input for the last XOR gate. This gate controls this registers input, because the output of the XOR goes to the D input for the rightmost flip flop in this register.
3- The first XOR gate we discussed does not have an output attached to anything yet. We will now proceed to attach it to two things, the input of the third register, with the intent of beginning to decode the garbage output of this XOR, as well having this XORs attached to a third XORs input, with its other input being attached to the Q of the rightmost bit of the second register mentioned in step 2. The set inputs for this third register should all be grounded, except for the rightmost bit, which will be set to the NOT (using a NOT gate) of the common reset switch.
4- This 3rd XOR's output is attached to our final shift register, this one I used a LogicWorks native shift register with a common clock input and SI input, which the XOR gate will go

to because this is the shifted D input essentially. This register is attached to four binary probes which will show us the decoded message.

Results:

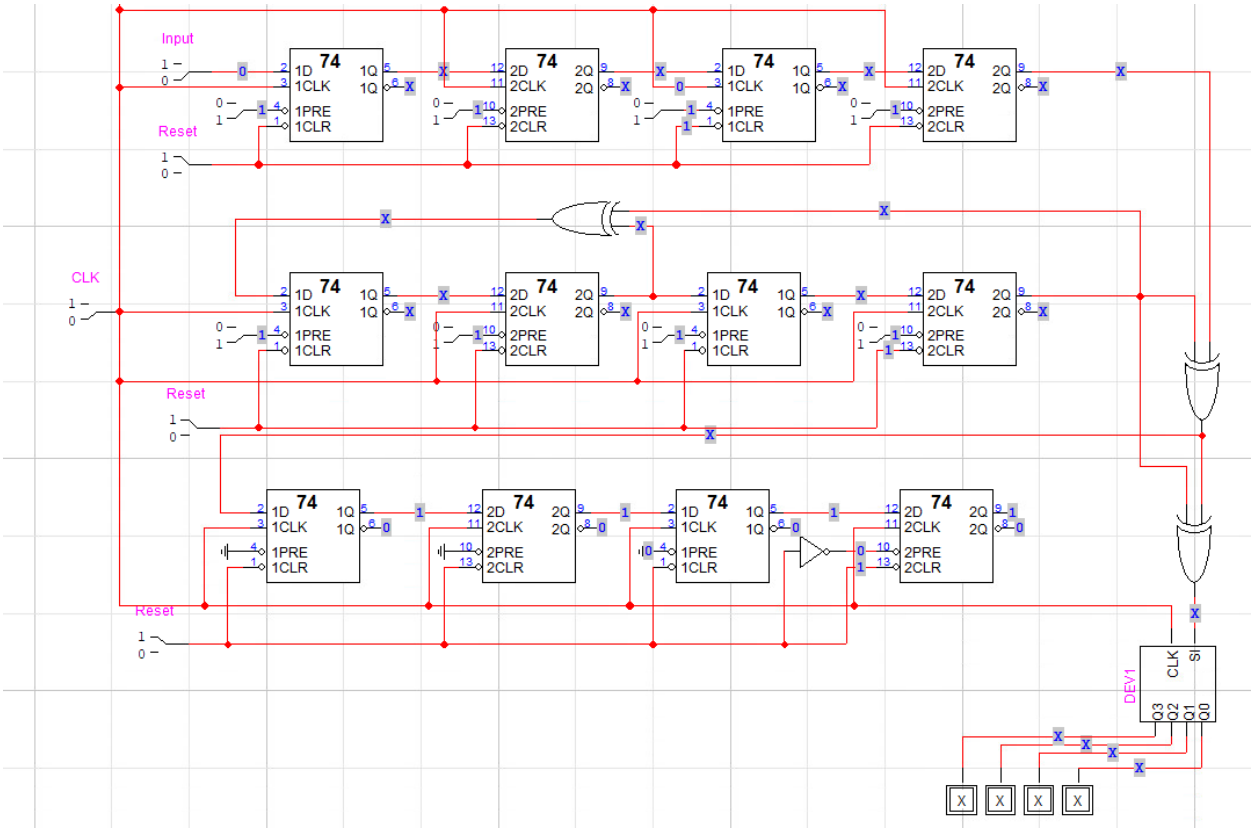

Figure 1: This is the diagram of my circuit. Notice the total of 4 4-bit shift registers, the three XOR gates, the common clock, and the 4 output binary probes. There is also the NOT on the LSB of the 3rd register down.

I tested for the three BCD values of 0001 or 1 in decimal, 0101 or 5, and 1001 or 9. After resetting all of the registers, inputting the number and then running the clock for another few cycles, I confirmed that all of these values displayed in the output of my final register. For the case of 0011 or 3, I confirmed the PRBS worked as well, both with serial input or directly inputting into the first register. The state I used as a beginning state for this testing was 0001, because it was the state after resetting all of the registers and running the clock.

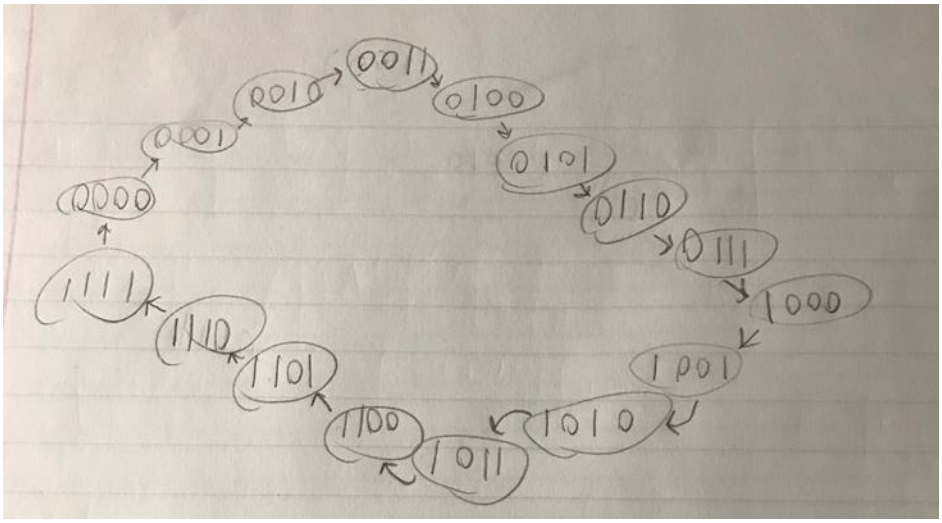

Figure 2: This is the state diagram for this lab. While I did not include the input and output values on the arrows, I wanted to mention them in this annotation. All of the visible arrows are caused by a 1 input, while a 0 input for any of the states would result in a loop onto the previous state, thus resulting in 2 outputs for each state. There are 16 states here so we can say that all the states are included.

Discussion:

Since there were not very many results of this lab aside from the construction of the circuit and verification that it works, there is not very much to discuss regarding the results. I would say that the majority of my questions and concerns about this lab occurred during the construction period, since conceptually understanding how a PRBS works is only part of the way to understanding how to create one. Learning what all of the inputs and outputs for each of the input bits was something very important. Knowing how to test the circuit had a bit of a learning curve as well, since initially I didn't know what positions to put the SET and RESET switches on the D flip flops, but then I learned that we decode this circuit by running the clock an extra few

cycles. I also believe that while the requirement wasn't to have 0000 in a 2^n state implementation of the PRBS, I believe mine did because there are 4 registers and 16 states.

Conclusion:

My final thoughts on this lab definitely reflect my opinion that this is a lab that helps to teach and reinforce a practical use for shift registers that we have used so much. PRBS is a concept that is applied in computer science and engineering applications such as scrambling often, and being able to experience and create a device like this is valuable for any future work that I may do with the same or similar devices. Modern society also puts an interesting tone on a project like this, because of media portraying encryption and decryption of code in a very polarizing light, like James Bond books/film. Learning the basics of a system like PRBS with a fairly small barrier for entry, but the potential of adding more XOR gates to make it a more complicated sequence is a trait of the sequence that makes it useful even today.