SOFTWARE 6:
PRBS ENCRYPTION & DECRYPTION

GENERAL RUBRIC

DEMO RUBRIC

In this lab, the student must create, in LogicWorks, a circuit capable of encoding and decoding a 4-bit message by transmitting each bit independently. The PRBS should work as indicated in the relevant module and in the instructions. Specifically, students will…

1. Use four D flip-flops to make a 4-bit register. This is register A, and it contains the data to be transmitted. The data is likely to be loaded in parallel.
2. Another 4-bit register with flip-flops labelled WXYZ performs the PRBS encryption. After each clock, W will become X XOR Z and everything else is shifted right.
3. Each message bit is Ao (last bit of register A) XOR Z (from the PRBS encryptor).
4. Make a decoder which is functionally the same as the encryptor, and should be exactly synchronized with the encryptor. To get the original, unencrypted message bit, do M (encrypted bit) XOR Z (from the PRBS decryption register).
5. The output should be stored in another 4-bit register.

Completion Requirements:

- ✓ The LogicWorks mock-up includes all four sets of registers and includes binary probes showing the values for all flip-flops, as well as the encrypted message bit. Alternatively, uses Simulation > Show Values.
- ✓ The data from register A is properly transmitted when the PRBS registers are initialized to some random non-zero 4-bit sequence.

REPORT RUBRIC

Scoring (out of 3 points):

- ✓ **[0.6 points]** Theory:
  - [0.3] Briefly detail your understanding of how PRBS generators work and for what applications they might be used.
  - [0.3] Long strings of randomly selected bits can be used as keys to XOR an entire message, making it appear entirely random. This method is known as One Time Pad (OTP). The problem is that the "pad" must be as long or longer than the message itself, making it absurdly long. What advantage does PRBS have over OTP?

- ✓ **[1.4 points]** Deliverables:
  - [0.8] Create two tables. In the first table, select a BCD value 1 through 9 and initialize the PRBS registers to `0001.` Transmit the message. For the initial state

and each clock, document the current state of register A, the output register, one of the PRBS registers (they should be the same), and the value of the encrypted bit. In the second table, do the same, but with the PRBS registers initialized to `0011`.

- [0.6] Create a transition table for the PRBS generator to show that all states (expect `0000`) are included.

✓ **[0.4 point]** Discussion section should adhere to the standard requirements.

✓ **[0.6 points]** Question:

- Why can't we include the all-zeroes (`0000`) state in our PRBS generator? Can you think of design that would allow us to include this state? Address any potential shortcomings your suggestion has.