Lottery System.



10-BIT PSEUDO RANDOM NUMBER GENERATOR.

NAME: Abhijeet R Deshmukh

MIS: 111909002

For lottery system we need an electronic device which will give us random numbers. So, our focus in this project is creation of random number generator.

# INTRODUCTION

**Random number generation** is a process which, often by means of a **random number generator** (**RNG**), generates a sequence of [numbers](https://en.wikipedia.org/wiki/Number) or [symbols](https://en.wikipedia.org/wiki/Symbol) that cannot be reasonably predicted better than by a [random](https://en.wikipedia.org/wiki/Random) chance. Random number generators can be truly random [*hardware random-number generators*](https://en.wikipedia.org/wiki/Hardware_random_number_generator) (HRNGS), which generate random numbers as a function of current value of some physical environment attribute that is constantly changing in a manner that is practically impossible to model, or [pseudorandom number generators](https://en.wikipedia.org/wiki/Pseudorandom_number_generator) (PRNGS), which generate numbers that look random, but are actually deterministic, and can be reproduced if the state of the PRNG is known.

In this project we will be focusing on PRNG.

# LFSR

We are using LFSR model of Pseudo random number generators.

In [computing](https://en.wikipedia.org/wiki/Computing), a **linear-feedback shift register** (**LFSR**) is a [shift register](https://en.wikipedia.org/wiki/Shift_register) whose input bit is a [linear function](https://en.wikipedia.org/wiki/Linear#Boolean_functions) of its previous state.

The most used linear function of single bits is [exclusive-or](https://en.wikipedia.org/wiki/Exclusive-or) (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value.

The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a [well-chosen feedback function](https://en.wikipedia.org/wiki/Primitive_polynomial_(field_theory)) can produce a sequence of bits that appears random and has a [very long cycle](https://en.wikipedia.org/wiki/Maximal_length_sequence).

## **Fibonacci LFSRs[**[**edit**](https://en.wikipedia.org/w/index.php?title=Linear-feedback_shift_register&action=edit&section=1)**]**

[](https://en.wikipedia.org/wiki/File:LFSR-F16.svg)

A 16-bit [Fibonacci](https://en.wikipedia.org/wiki/Fibonacci) LFSR. The feedback tap numbers shown correspond to a primitive polynomial in the table, so the register cycles through the maximum number of 65535 states excluding the all-zeroes state. The state shown, 0xACE1 ([hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal)) will be followed by 0x5670.

The bit positions that affect the next state are called the taps. In the diagram the taps are [16,14,13,11]. The rightmost bit of the LFSR is called the output bit. The taps are XOR'd sequentially with the output bit and then fed back into the leftmost bit. The sequence of bits in the rightmost position is called the output stream.

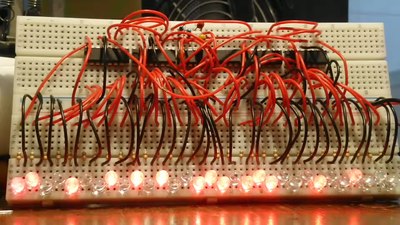
* The bits in the LFSR state that influence the input are called *taps*.
* A maximum-length LFSR produces an [m-sequence](https://en.wikipedia.org/wiki/Maximum_length_sequence) (i.e., it cycles through all possible 2*m* − 1 states within the shift register except the state where all bits are zero), unless it contains all zeros, in which case it will never change.
* As an alternative to the XOR-based feedback in an LFSR, one can also use [XNOR](https://en.wikipedia.org/wiki/XNOR).[[2]](https://en.wikipedia.org/wiki/Linear-feedback_shift_register#cite_note-2) This function is an [affine map](https://en.wikipedia.org/wiki/Affine_transformation), not strictly a [linear map](https://en.wikipedia.org/wiki/Linear_map), but it results in an equivalent polynomial counter whose state is the complement of the state of an LFSR. A state with all ones is illegal when using an XNOR feedback, in the same way as a state with all zeroes is illegal when using XOR. This state is considered illegal because the counter would remain "locked-up" in this state. This method can be advantageous in hardware LFSRs using flip-flops that start in a zero state, as it does not start in a lockup state, meaning that the register does not need to be seeded in order to begin operation.

The sequence of numbers generated by an LFSR or its XNOR counterpart can be considered a [binary numeral system](https://en.wikipedia.org/wiki/Binary_numeral_system) just as valid as [Gray code](https://en.wikipedia.org/wiki/Gray_code) or the natural binary code.

The arrangement of taps for feedback in an LFSR can be expressed in [finite field arithmetic](https://en.wikipedia.org/wiki/Finite_field_arithmetic) as a [polynomial](https://en.wikipedia.org/wiki/Polynomial) [mod](https://en.wikipedia.org/wiki/Modular_arithmetic) 2. This means that the coefficients of the polynomial must be 1s or 0s. This is called the feedback polynomial or reciprocal characteristic polynomial. For example, if the taps are at the 16th, 14th, 13th and 11th bits (as shown), the feedback polynomial is

{\displaystyle x^{16}+x^{14}+x^{13}+x^{11}+1.}

The "one" in the polynomial does not correspond to a tap – it corresponds to the input to the first bit (i.e. *x*0, which is equivalent to 1). The powers of the terms represent the tapped bits, counting from the left. The first and last bits are always connected as an input and output tap respectively.



A Fibonacci 31bit linear feedback shift register with taps at positions 28 and 31, giving it a maximum cycle and period at this speed of nearly 6.7 years. The circuit uses 4x74HC565N for the shift registers, a 74HC86N for the XOR and an inverter, and an LMC555 timer for clock pulses.

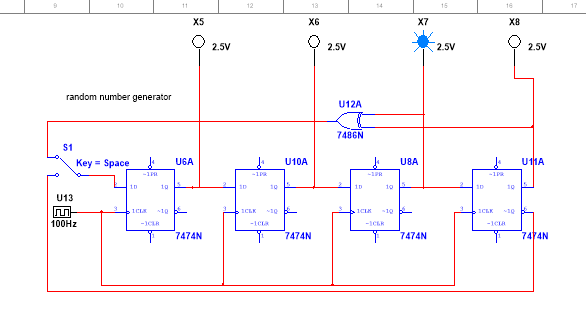
The LFSR is maximal-length if and only if the corresponding feedback polynomial is [primitive](https://en.wikipedia.org/wiki/Primitive_polynomial_(field_theory)). This means that the following conditions are necessary (but not sufficient):

* The number of taps is [even](https://en.wikipedia.org/wiki/Even_and_odd_numbers).
* The set of taps is [setwise co-prime](https://en.wikipedia.org/wiki/Coprime_integers" \l "Coprimality_in_sets" \o "Coprime integers); i.e., there must be no divisor other than 1 common to all taps.

Tables of primitive polynomials from which maximum-length LFSRs can be constructed are given below and in the references.

There can be more than one maximum-length tap sequence for a given LFSR length. Also, once one maximum-length tap sequence has been found, another automatically follows. If the tap sequence in an *n*-bit LFSR is [*n*, *A*, *B*, *C*, 0], where the 0 corresponds to the *x*0 = 1 term, then the corresponding "mirror" sequence is [*n*, *n* − *C*, *n* − *B*, *n* − *A*, 0]. So the tap sequence [32, 22, 2, 1, 0] has as its counterpart [32, 31, 30, 10, 0]. Both give a maximum-length sequence.

# 4bit PRNG



Here the Q3 and Q4 are XORed to give the feedback to D1 and based on the given input shift register will load the input in SIPO manner and we will 0-15 …total 15 different random numbers unless we give input =0000.

D1 = Q3(+) Q4 …. (+) denotes XOR gate.

When some initial input is given then circuit work as shift register as usual last number (Q4) is thrown out and Q1 is introduced as XORed input and other will shift by one position (i.e., D2=D1-, D3=D2- , D4=D3-)

Truth table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decimal output | Q1=D1=Q3(+) Q4 (of previous input) | D2=Q2 | D3=Q3 | D4=Q4 |
| (initial input=1) | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 9 | 1 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 8 (restart of sequence) | 1 | 0 | 0 | 0 |

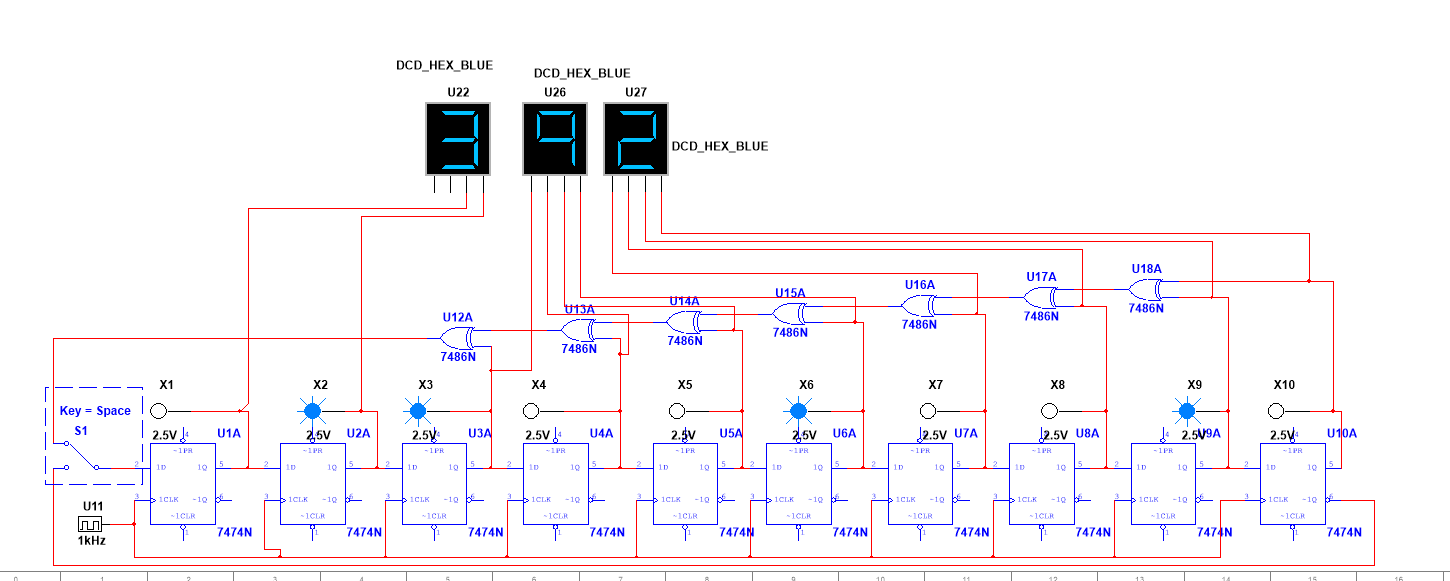
Based on given initial input we get different set of 15 random numbers but after 15 outputs pattern will repeat. This same phenomenon is used to create 10bit PRNG but here twist is number of XOR gates and the position of Qs from where XOR is taken determines the longest cycle of random numbers. But applying any number of XOR gates also satisfies condition of randomness only thing that will change is total number of outputs.

For 10bit PRNG it is recommended to use 10th and 7th position to attach XNOR and get feedback from them for getting longest cycle. But here we have used all 3-10Qs for XOR feedback and we are getting a good random pulse as shown in timing diagram.

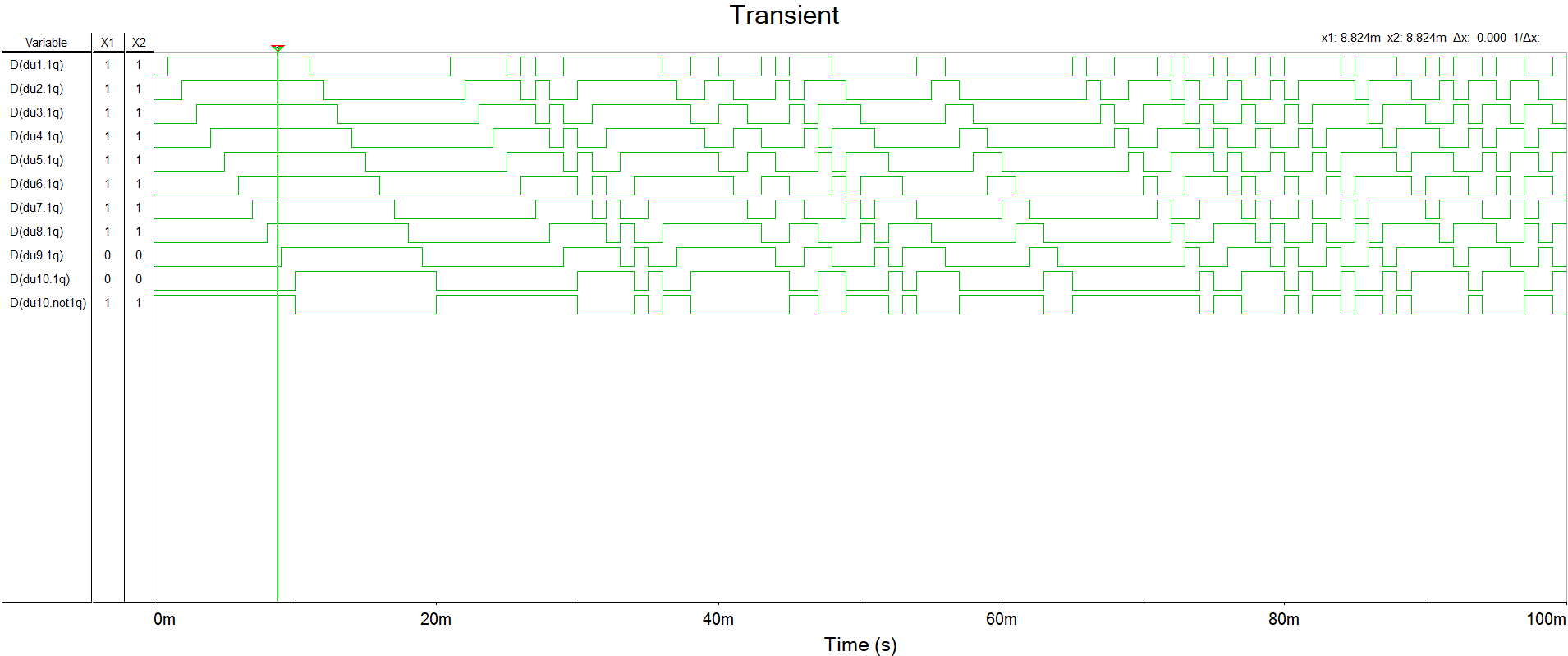
# Components

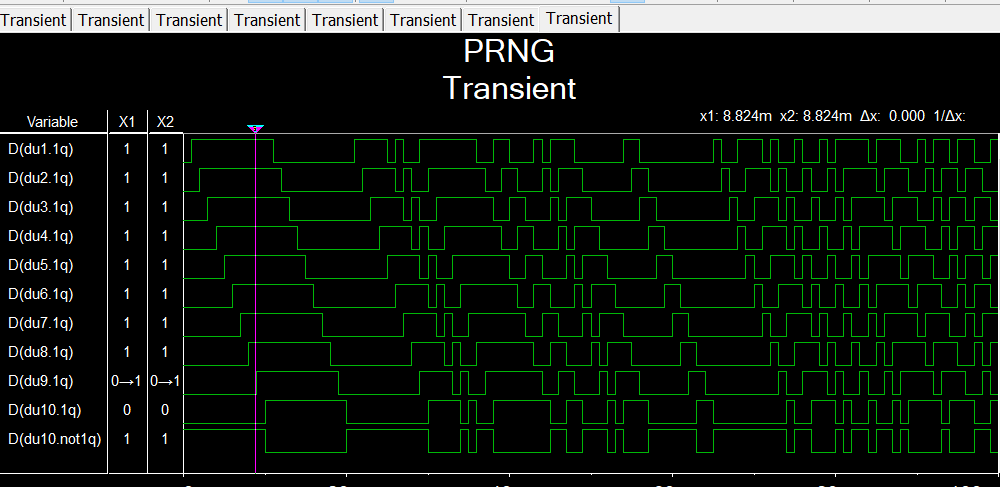
D-FF, XOR gate, HEX DISPLAY OR (10bit binary to BCD convertor (74ls185 ICs+ cascading knowledge) + seven segment display + BCD to 7 segment display decoder 7447ic), LEDs, clock.

# Design 10BIT PRNG for LOTTERY SYSTEM



# Observation





# Conclusion:

PRNG can be created using basic LFSR circuits and a lottery system can be created from that, one can also create python or c programme for MERSENNES TWISTER algorithm to implement 2^1897 -1 random number generator.

Special thanks to faculty Uttam Chaskar