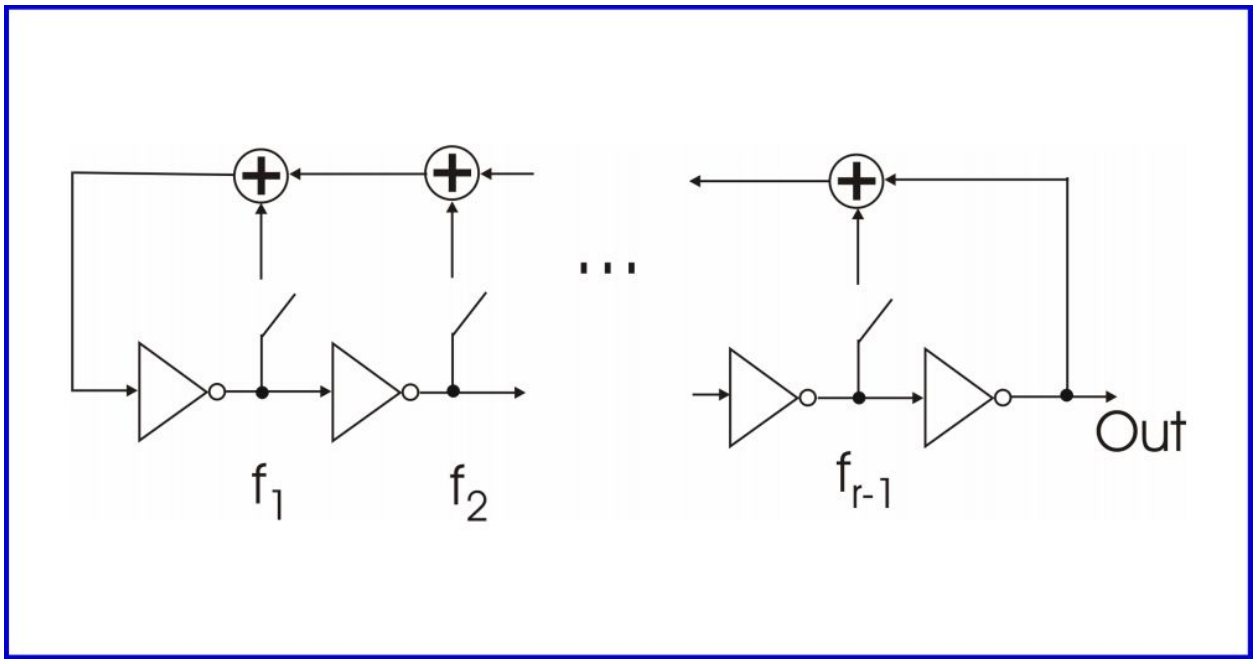
DSD Project

RNG using a Fibonacci Inverter Ring Oscillator



Agents of Chaos

3/10/2016

V. V. Shashank - ES16BTECH11025

Jeel Bhavsar - ES16BTECH11005

Akshita Ramya - ES16BTECH11012

Aravind Ganesh - EE16BTECH11026

Bhavya Sri - EE16BTECH11020

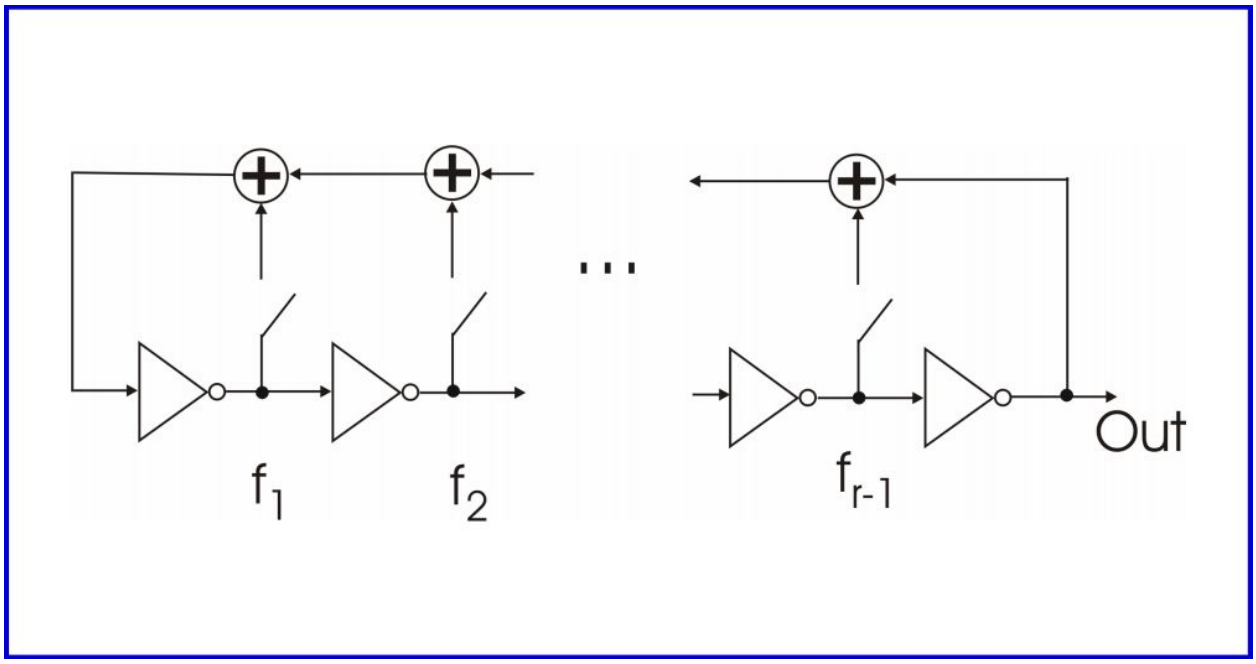
INTRODUCTION

Though not immediately apparent, Random number generators are useful to a degree beyond comprehension. Random number generators have applications in gambling, statistical sampling, computer simulation, cryptography, completely randomized design, and other areas where producing an unpredictable result is desirable. Generally, in applications having unpredictability as the paramount, such as in security applications, hardware generators are generally preferred over pseudo-random algorithms, where feasible.

For our project, we have implemented a hardware random number generator using only logic gates and sequential circuits. We have done this by implementing a simple structure of a Fibonacci Ring Oscillator (FIRO), which has been extensively described in [1]. In theory, it is only a pseudo random number generator, but in reality, it's properties are closer to that of a true RNG due the metastability of states, caused by the delays in the oscillator.

Fibonacci Ring Oscillator

A Fibonacci ring oscillator is a modification of the default ring oscillator, where the feedback is more complicated by the incorporation of a lot XOR gates, in a way which is similar to that of the Fibonacci configuration of an LFSR.



The feedback connections are chosen in a way so as to satisfy the condition that the oscillator has no resting points. Such feedback connections can be characterized nicely by a simple polynomial $f(x)$, such that

$$f(x) = (1+x)\,h(x)$$

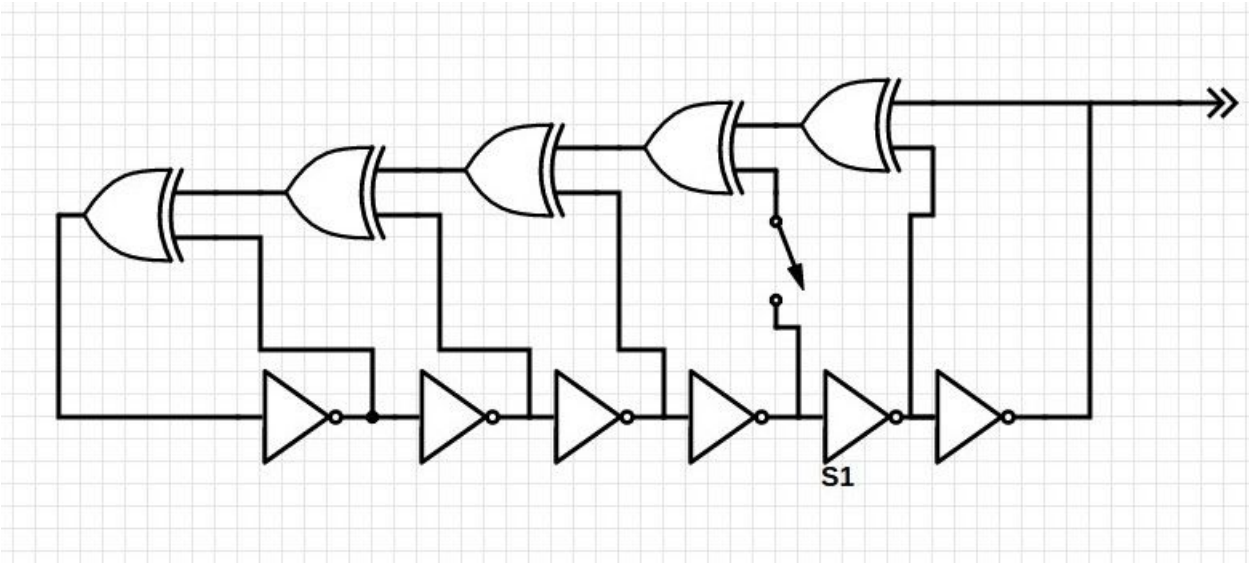
Where $h(x)$ is a polynomial such that $h(-1) = 1$ *and* $h(x)$ *is not divisible by* $(1+x)$. Primitive polynomials[2] act as nice candidates for $h(x)$ as they satisfy these properties by default, and are well documented. It is also found that they produce better pseudorandom properties than other polynomials. The primitive polynomial of our choice was

$$h(x) = 1 + x^2 + x^5,$$

which yields an $f(x)$ of

$$f(x) = 1 + x + x^2 + x^3 + x^5 + x^6$$

What this function implies is that the switches f_i, each of which correspond to the powers of x in the polynomial, are to be *on.* This has been shown below -



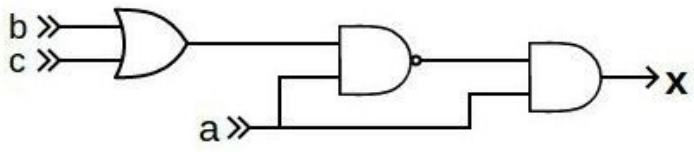
Converting the serial stream to a 4-bit number

We used a 4-bit universal bidirectional register (74194) to implement serial input parallel output. We are using the register in serial input (from ring oscillator) and 4-bit parallel output mode (SIPO Mode). The clock input is received from the arduino board. The serial data input is received from the random number generator.

Since we were using only one 7-segment display we cannot display numbers greater than 9 or $(1001)_2$. The parallel outputs that the display would receive would be from 0000 to 1111 in binary. So we came up with a logic to avoid getting numbers greater than 9.

If our number is 'abcd' in binary with 'a' being the most significant digit,

$$x = \overline{a \cdot (b + c)} \cdot a$$



New number = xbcd

According to this logic,

Case 1: a=0, b or c or both are 1

There is no issue with this since the number will be less than 9. The resulting x = 0

Case 2 : a=1, b or c or both are 1

This the case for which our logic has been designed. The resulting x becomes 0 thus our new number xbcd will be less than 9.

Sampling and display

The value taken from SIPO has high frequency;. If it was intended to give this as an input to a 7-segment display through a BCD(7447) it would be too fast for the human to observe or detect any variation in display.

So, we used D-flipflops (7474) to store the bits for a while to display. And also we also included a push-button to control manually the sampling. We take random samples of any 4 consecutive bits generated by the RNG**.** The D-flipflops PRESET and CLEAR are set to high and positive edge triggering changes the value stored in the flip-flops. Now when using a push-button the clock pulse is made low the bits are retained in the flip-flops which are in turn given to the 7447 BCD via which the 7-segment display actually display a number in decimal system.

Tests

- Arduino test - Printing the output bits of the RNG on the serial monitor, we can observe the random sequence of zeros and ones.

Debugging

- First task was to find suitable components and to find alternatives for them if they were not available in the lab.
- We tested all the ICs using an LED.
- Arduino as a power source wasn't sufficient to power all the circuit we made. So, we we used an FTDI cable and plugged it into a 5-volt phone charger for power supply.
- We also had to check whether the bread boards were functioning properly and found a breadboard was faulty.
- Found a suitable resistance for connecting to the 7-segment display.

Citations

[1] - J. Dj. Goli ć , "New Methods for Digital Generation and Postprocessing of Random Data," IEEE Trans. Computers, vol. 55(10), pp. 1217-1229, Oct. 2006

[2] - Primitive Polynomials on Wolfram

Contributions

Research - Shashank

Designing and constructing FIRO - Aravind, Shashank, Akshita

SIPO and modulo 9 circuit - Jeel

D-flipflop and push button circuit - Akshita and Jeel

7447 BCD and 7 - segment display circuit - Akshita and Bhavya

Testing and debugging - Each for his/her own circuit

Coding clock and debugging modules on Arduino - Shashank, Aravind

Documentation - Everyone for their own circuits and contributions, compiled and formatted by Bhavya