Oh The Humanity! What Have You Done To The MLU!?

MLU Functions in the Tritium Cycle

Evan McClellan

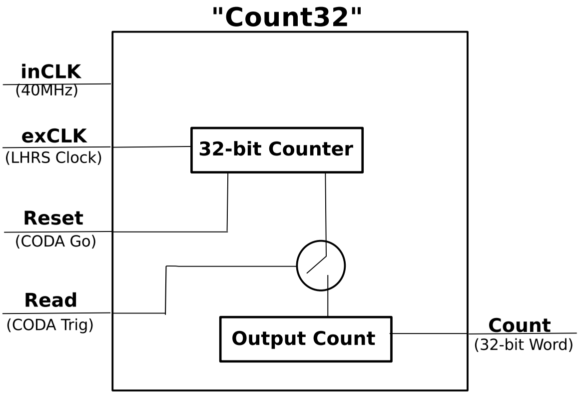
February 8, 2018

Fast Clock Counter

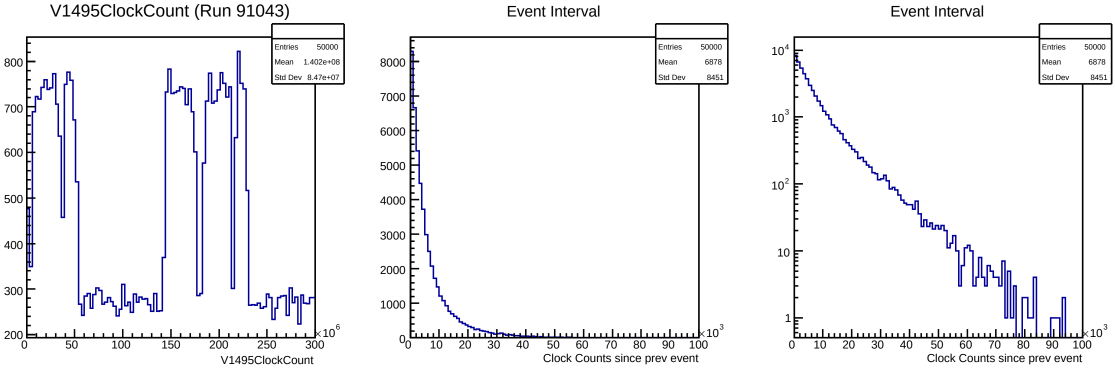
- Count LHRS clocks since start of run
- Read-out every physics event
- LHRS clock frequency = 103 kHz
- Two-Arm event syncing, inter-event duration, etc.

Register Size

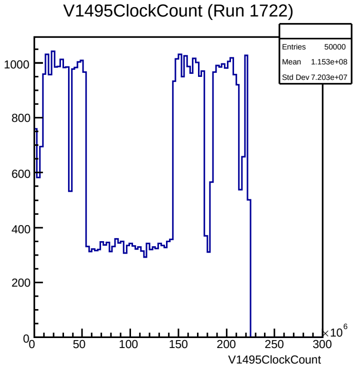
nbits	Period
16	0.64 s
32	11.6 hr
48	86.7 yr
64	5.68 Myr

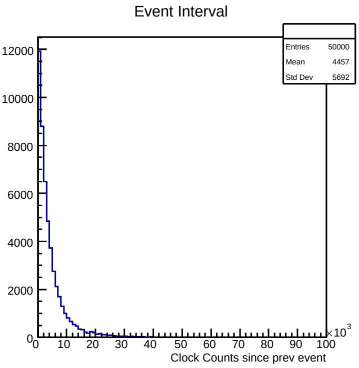


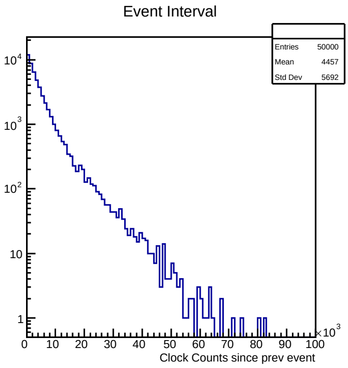
Clock Counter Data (RHRS)



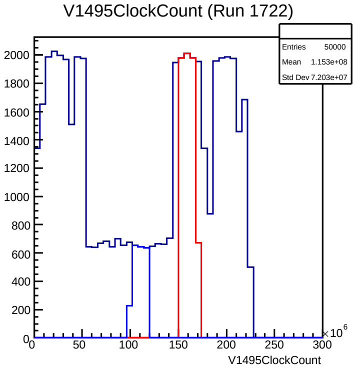
Clock Counter Data (LHRS)

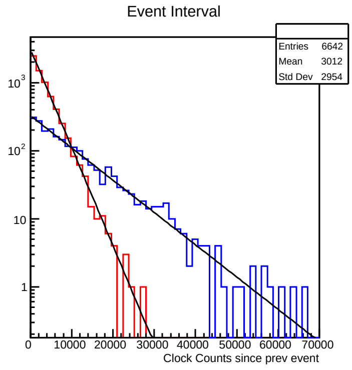


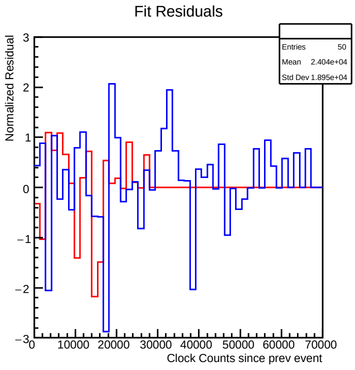




Clock Counter Data (LHRS): Beam vs Cosmics







A Random Pulser in Programmable Logic

Linear Feedback Shift Registers, Inverse Transform Sampling, and Rational Function Fitting, Oh My!

Evan McClellan

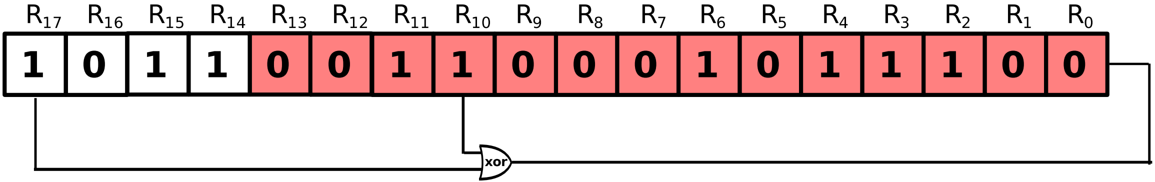
February 8, 2018

Why make a Fast Random Trigger?

Accurate deadtime vs rate mapping without beam
(less accurate if read-out is zero-suppressed)

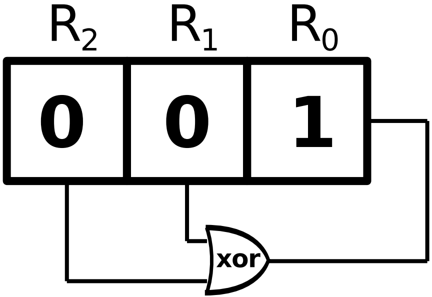
What is a Linear-Feedback Shift Register?

Pseudo-random number generator.
Easy to implement in hardware (programmable logic)
Used for security in GSM phones.



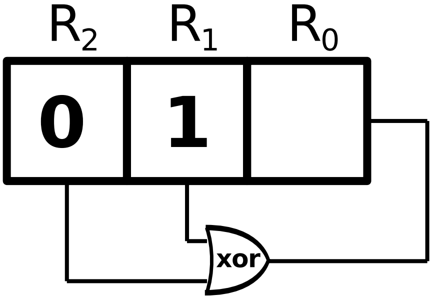
LFSR 3-bit Demo

R = '001' = 1

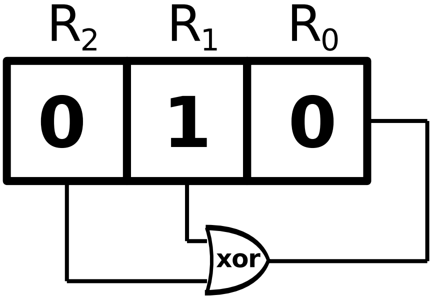


LFSR 3-bit Demo

R = '001' = 1



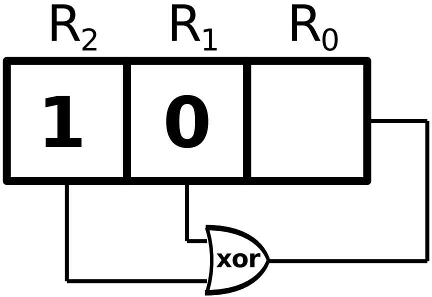
LFSR 3-bit Demo



R = '001' = 1

R = '010' = 2

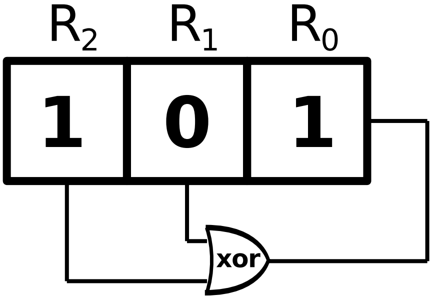
LFSR 3-bit Demo



R = '001' = 1

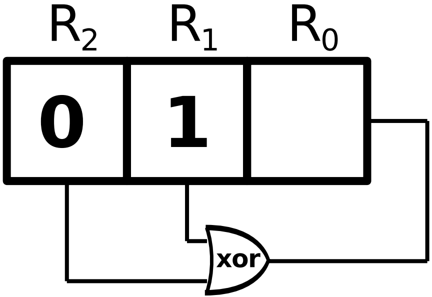
R = '010' = 2

LFSR 3-bit Demo



R = '001' = 1
R = '010' = 2
R = '101' = 5

LFSR 3-bit Demo

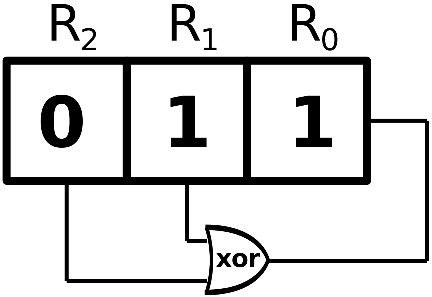


R = '001' = 1

R = '010' = 2

R = '101' = 5

LFSR 3-bit Demo



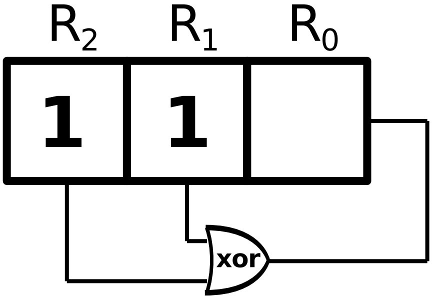
R = '001' = 1

R = '010' = 2

R = '101' = 5

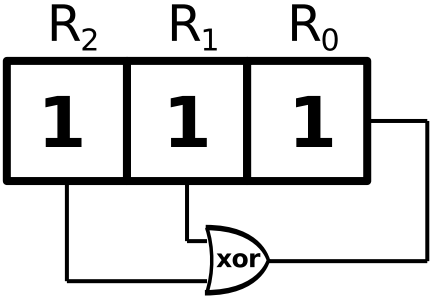
R = '011' = 3

LFSR 3-bit Demo



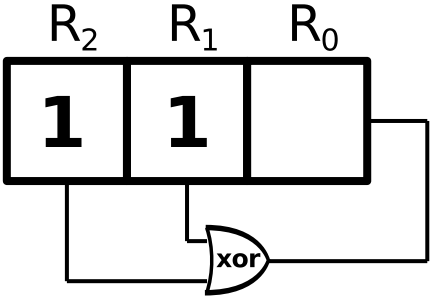
R = '001' = 1
R = '010' = 2
R = '101' = 5
R = '011' = 3

LFSR 3-bit Demo



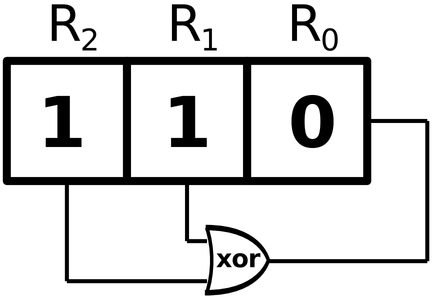
R = '001' = 1
R = '010' = 2
R = '101' = 5
R = '011' = 3
R = '111' = 7

LFSR 3-bit Demo



R = '001' = 1
R = '010' = 2
R = '101' = 5
R = '011' = 3
R = '111' = 7

LFSR 3-bit Demo



R = '001' = 1

R = '010' = 2

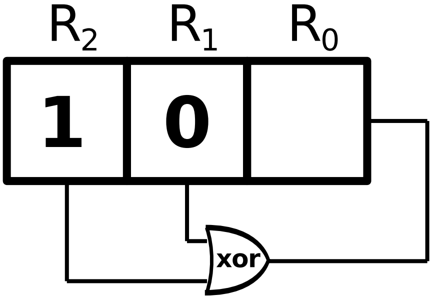
R = '101' = 5

R = '011' = 3

R = '111' = 7

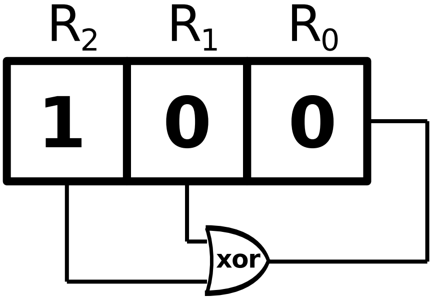
R = '110' = 6

LFSR 3-bit Demo



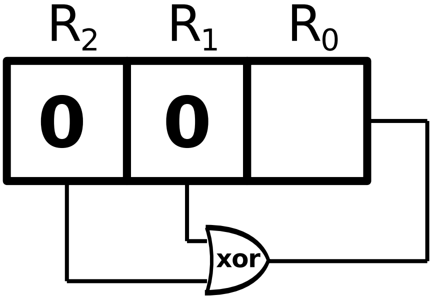
R = '001' = 1
R = '010' = 2
R = '101' = 5
R = '011' = 3
R = '111' = 7
R = '110' = 6

LFSR 3-bit Demo



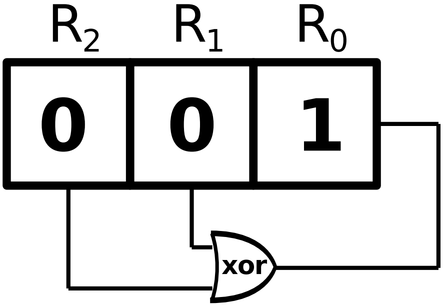
R = '001' = 1
R = '010' = 2
R = '101' = 5
R = '011' = 3
R = '111' = 7
R = '110' = 6
R = '100' = 4

LFSR 3-bit Demo



R = '001' = 1
R = '010' = 2
R = '101' = 5
R = '011' = 3
R = '111' = 7
R = '110' = 6
R = '100' = 4

LFSR 3-bit Demo

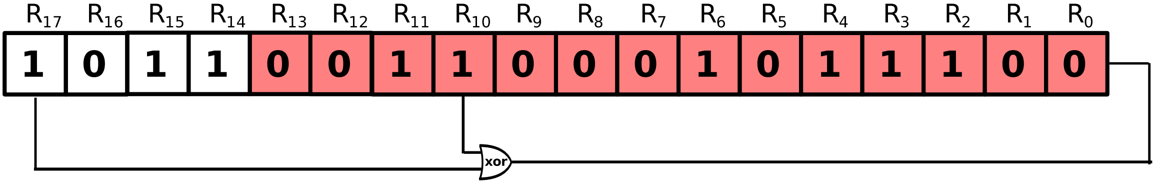


R = '001' = 1
R = '010' = 2
R = '101' = 5
R = '011' = 3
R = '111' = 7
R = '110' = 6
R = '100' = 4
R = '001' = 1

13 bits and 18 bits

18-bit LFSR, lowest 13 bits used as pRNG

output range [0,8191], cycle repeats every 262143 cycles ($= 6.6$ milliseconds)



What do intervals of stochastic events look like?

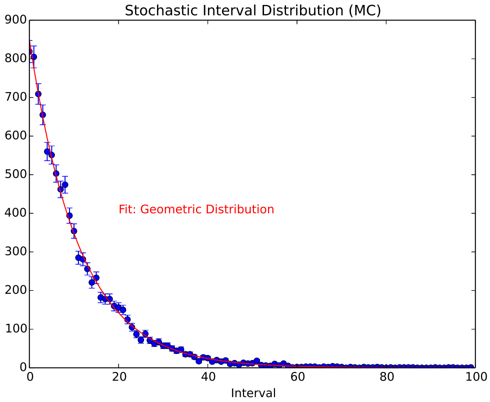
Exponential Distribution (continuous)

$$f_{pdf}(x) = \lambda e^{-\lambda x}$$

Geometric Distribution (discrete)

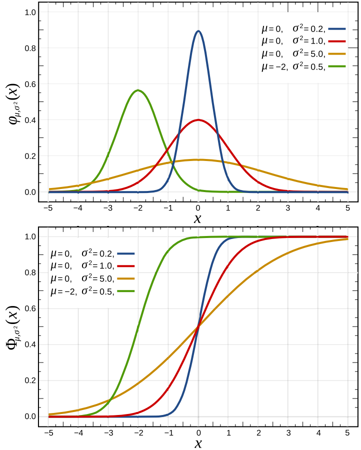
$$g_{pdf}(k) = (1 - p)^k p$$

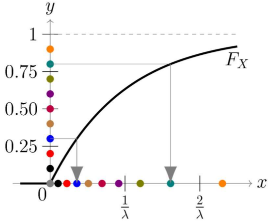
- p: Probability of success
- k: number of failed trials



From Uniform pRNG to Stochastic Interval (Overkill)

Inverse Transform Sampling





$$f_{cdf}(x) = \int_{-\inf}^{x} f_{pdf}(t)dt$$

$$y = f_{cdf}(x)$$

$$x = f_{cdf}^{-1}(y)$$

From Uniform pRNG to Stochastic Interval (Overkill)

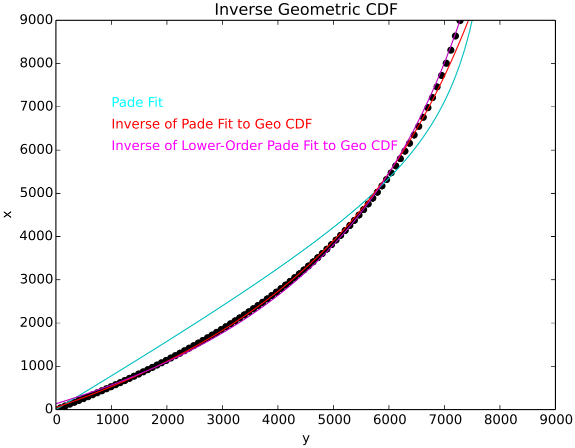
Exponents are Hard; Rational Functions are Easy!

$$g_{pdf}(k) = (1-p)^k p$$

$$g_{cdf}(k) = 1 - (1-p)^{k+1}$$

$$p_1(k) = \frac{a + bk}{1 + dk}$$

$$p_2(k) = \frac{a + bk}{8192 + 2k}$$



From Uniform pRNG to Stochastic Interval (Overkill)

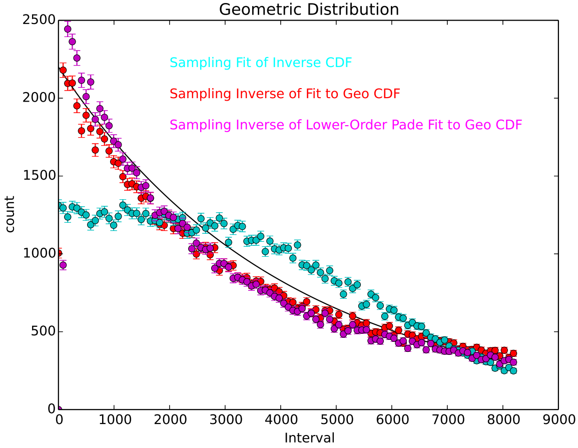
Exponents are Hard; Rational Functions are Easy!

$$g_{pdf}(k) = (1-p)^k p$$

$$g_{cdf}(k) = 1 - (1-p)^{k+1}$$

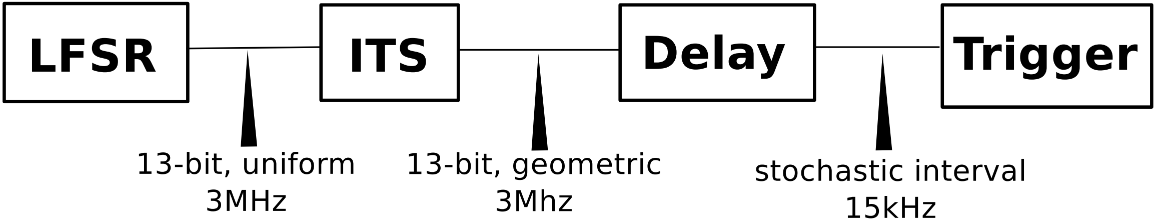
$$p_1(k) = \frac{a + bk}{1 + dk}$$

$$p_2(k) = \frac{a + bk}{8192 + 2k}$$



From Uniform pRNG to Stochastic Interval (Overkill)

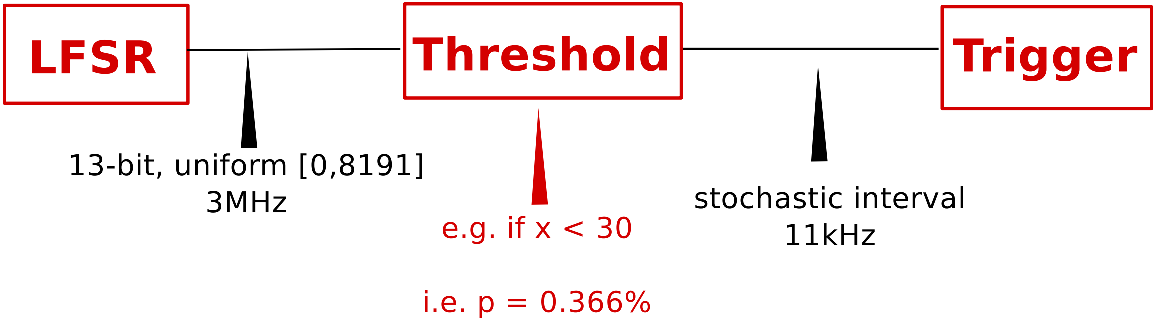
Putting It All Together



(In principle, this techique could generate *any* output distribution.)

From Uniform pRNG to Stochastic Interval (Jedi Solution)

Real-Time Stochastic Trigger Generation



The End

Thanks!