Randomness extractors embedded with physical random number generators

July 10, 2015

1 Abstract

We describe a series of Randomness Extractors for removing bias and residual correlations in random numbers generated from measurements on noisy physical systems. The structures of the randomness extractors are based on Linear Feedback Shift Registers (LFSR). This leads to a significant simplification in the implementation of randomness extractors.

2 Drawings

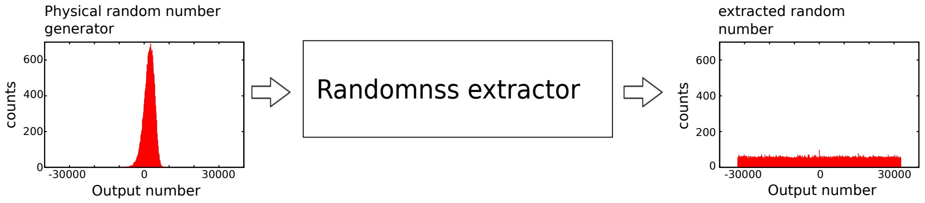

Figure 1: flowchart of the process

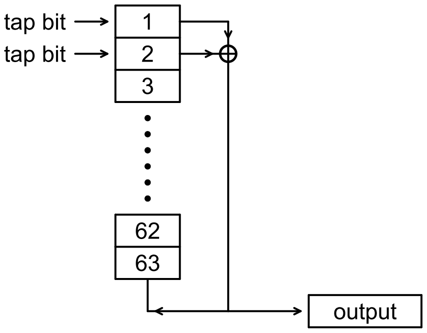

Figure 2: A 63-bit Fibonacci-type Linear Feedback Shift Register

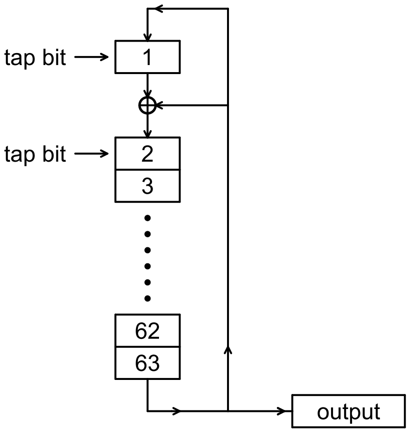

Figure 3: A 63-bit Galois-type Linear Feedback Shift Register

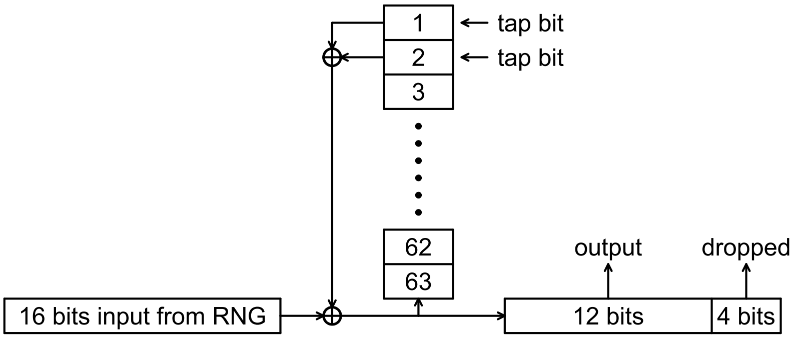

Figure 4: Example of a Fibonacci-type randomness extractor, serial input/output

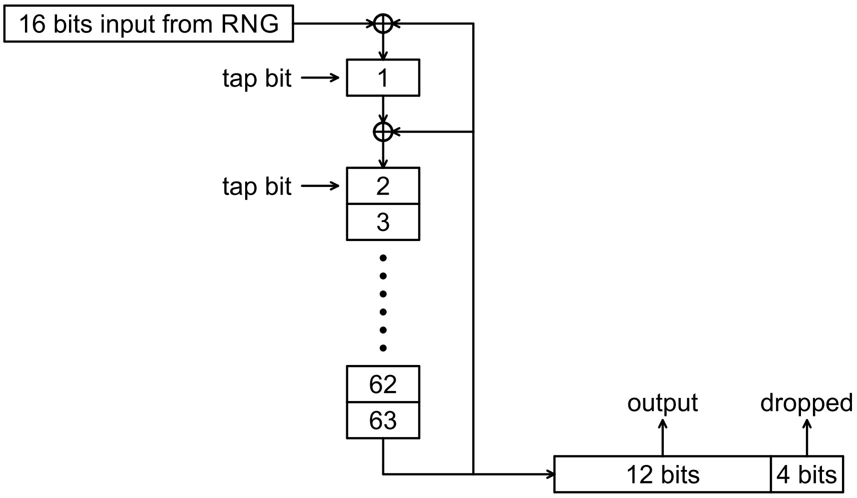

Figure 5: Example of a Galois-type randomness extractor, serial input/output

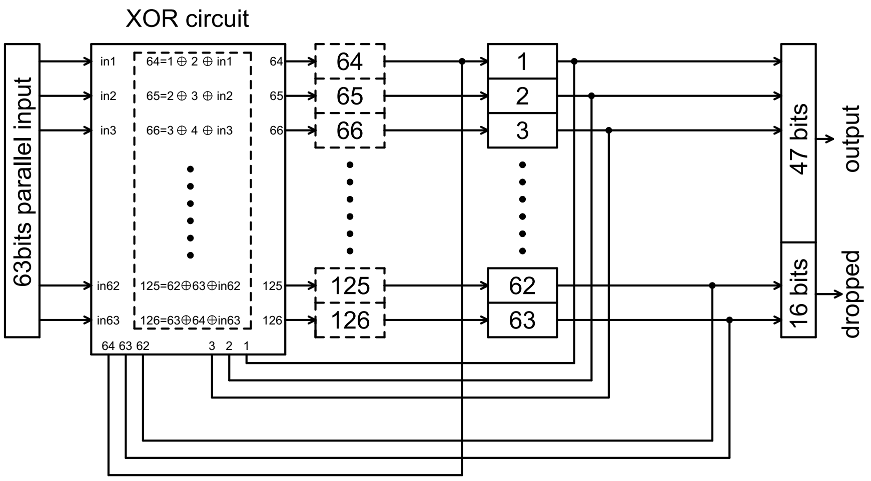

Figure 6: Example of a Fibonacci-type randomness extractor, parallel input/output

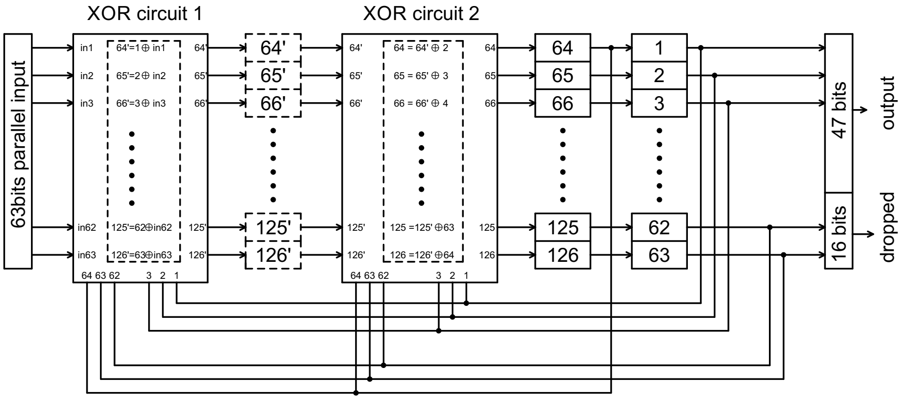

Figure 7: Example of a Fibonacci-type randomness extractor, parallel input/output

3 Invention background

The present invention is a randomness extraction processes that converts biased random data into unbiased random bits.

3.1 Random number generation

A range of applications ranging from encryption to numerical modeling require random numbers to be generated at high speed.[1, 2, 3] In particular, when information security is of concern, the quality of the generated random numbers should have good statistical properties. Ideally, they should possess no bias or correlations. In other words these random numbers should be intrinsically unpredictable.

Random number generators can be categorized into either pseudo-random number generators or physical random number generators. Pseudo-random number generators are deterministic since they employ complex algorithmic functions to generate their output, thus their output cannot be considered intrinsically random. However, pseudo-random number generators are easy to implement and are often used for computational purposes such as Monte-Carlo Simulations.

For applications that require a source of high quality random numbers for example cryptographic key generation, a physical random number generator whose output is based on intrinsically unpredictable physical processes is preferable. Some typical sources of physical randomness used to generate a source of raw random numbers include the detection of emission from radioactive nuclei, key strokes derived from human input and various types electrical/optical noises.[1, 4, 5, 6] Measurements on sources of physical randomness are often slower than the computation speed of processors, and thus limits the output rate.

The raw random numbers derived from physical sources are also typically biased and cannot be used directly. A post-processing step (also known as randomness extraction) is required to ensure the quality of the output numbers.[6, 7] This post-processing generally involves performing logic operations on the raw random data and will introduce further time delay in the final output stage.

3.2 Randomness extraction

From an information-theory point of view, a randomness extractor is a process that converts a statistically weak binary stream into a new stream, with each bit in the output stream possessing one bit of Shannon entropy. The process should be in principle irreversible such that the raw data cannot be recovered from the output.

One famous example of such an extractor is the so called Von-Neumann extractor.[8] This specific extractor operates by grouping the raw input bit stream into pairs and maps them to individual bits. It has been shown that the Von-Neumann extractor eliminates the bias of the bit stream, but is not effective against correlations. Also, the Von-Neumann extractor is not efficient as the output rate

is reduced by more than half of the input rate.[7]
Another typical implementation of a randomness extractor uses a cryptographic hashing function.[2, 6] Such hashing functions are proven to be effective and secure, but the complexity of the hashing functions slows down the processing speed and increases the implementation difficulty.

A faster output rate generally requires more memory space and faster processing units, thus compact implementations of such algorithms are difficult to realize.

The applicant came to notice, while implementing the randomness extractors, another feature of importance, known as the conservation of entropy.

This proposition can be understood as the following: Since the randomness extractor is a deterministic process that does not input any additional entropy into the input stream, the entropy of the output stream should be conserved. From an information-theoretic point of view, each bit at the input of the randomness extractor contains less than 1 bit of Shannon entropy. For an ideal randomness extractor we require the output binary stream to possess a Shannon entropy of 1 per bit. One method of conserving entropy is to truncate the length of the output stream.

From a cryptographic angle, a shorter output stream requires dropping a certain portion of the output bits, which makes the entire extraction process practically irreversible. Potential attackers are unable to recover the raw random data and hence the security of the system is ensured.

The applicant observes that implementing such an entropy conversation process improves the output quality of the randomness extractors.

3.3 Prior arts

Prior arts to be quoted?(saw several other patents did that...)
some possible quotes
US 8219602
US 6581078
US 8364977

4 Summary of the disclosure

The applicant proposes an efficient implementation of a randomness extractor that processes raw random data from a source of physical randomness and produces high quality random numbers at the output.

The applicant observes that a simple and compact structure commonly known as a Linear Feedback Shift Register (LFSR) is suitable for randomness extraction due to properties such as spectrum whitening and uniformizing the distribution of its output stream.

The applicant proposes a modification to the LFSR structure such that instead of a deterministic structure , the LFSR receives a random serial input from a physical source, performs a series of logic operations and output an extracted random stream. The extracted random stream is at the same time used to update the internal state of the LFSR and prepare for the next round of operation.

The applicant proposes two different structures of randomness extractors based on the LFSR, namely the serial Fibonacci and serial Galois type.

The applicant also proposes two additional structures, namely the parallel Fibonacci and Galois type randomness extractors. The two parallel version extractors have much faster output rates compared to their corresponding serial versions, with an acceptable increase in computational resources.

The applicant employs the entropy conservation process mentioned earlier and applies a bit-dropping step to all the extractors introduced to ensure a shorter output stream and a conserved input/output entropy. The portion of dropped bits is decided by first statistically measuring the entropy of the raw random data, and then truncating the output stream length to match amount of entropy measured.

The applicant claims the proposed structure to be a high speed, efficient randomness extraction scheme, and can be implemented in compact and efficient designs. The proposed structures may possibly be implemented as part of on-chip random number generators for embedded systems.

5 Description of the drawings

Fig.1 shows a general flow chart of the random number generation process from a physical randomness source. The process includes a physical randomness source, a measurement/sampling unit, and a randomness extractor unit.
Fig.2 schematically shows the common structure of a Fibonacci-type Linear Feedback Shift Register.
Fig.3 schematically shows the common structure of a Galois-type Linear Feedback Shift Register.
Fig.4 shows a possible implementation of a serial version, Fibonacci-type randomness extractor
Fig.5 shows a possible implementation of a serial version, Galois-type randomness extractor
Fig.6 shows a possible implementation of a parallel version, Fibonacci-type randomness extractor
Fig.7 shows a possible implementation of a parallel version, Galois-type randomness extractor

6 Description of the invention

6.1 Randomness from physical sources

A physical random number generator (RNG) is based on measurements of a source of physical randomness. Different implementations of physical random number generators have been investigated. Typical sources of physical randomness include radioactive counters, human behaviours(key strokes, mouse movements, etc), electrical/optical noises, etc.[1, 4, 5, 6] A physical randomness source is usually quantum mechanical or chaotic, thus the measurement outcome is believed to be unpredictable.

An example of such a randomness source and measuement is shown in Fig.2. The applicant measures the noise from an optical detector to generate the raw random data. Optical noise is believed to originate from quantum mechanical fluctuations and thus effectively unpredictable. This is advantageous as the measurements can be performed at high speed and the detection unit is compact.

The outcome of the physical measurement is converted by an analog/digital convertor (ADC) into a binary number of certain bit length, determined by ADC. For a n-bit ADC, there are 2^n possible outcomes. The statistically weak raw data from the measurements will be distributed among these possible outcomes with a non-uniform distribution.

6.2 Shannon entropy of a randomness source

In this section we introduce the concept of Shannon entropy as an measure of the quality of the randomness source. A higher value of Shannon entropy indicates that the source is more random while a Shannon entropy of zero indicates that the source is perfectly predictable.

Before we perform the randomness extraction process, we characterize the amount of entropy of the source of randomness as measured with our n-bit ADC such that one may estimate the amount entropy to be retained later.

Each outcome of a measurement with a n-bit ADC can be considered as a random variable X with 2^n possible values. The Shannon entropy is defined as for a random variable X as
$H(X) = -\sum_{x_i} P(X = x_i) \log P(X = x_i)$
where $\sum_{x_i}$ indicates the sum over all possible values of X, and $P(X = x_i)$ is the probability of the variable X with value x_i.

The probability of each outcome $P(X = x_i)$ can be determined by counting its number of occurances over a large number of measurement outcomes. For a large enough sample size, we this formula is an accurate representation of the Shannon entropy of the source of randomness.

For a n-bit ADC, the value of $H(X)$ is between 0 and n, with 0 being perfectly non-random and n being perfectly random. A perfectly random case corresponds to a uniform distribution among all 2^n possibilities.

6.3 Linear Feedback Shift Register (LFSR)

A brief introduction of the so called Linear Feedback Shift Register is necessary before introducing the proposed randomness extractors.

A Linear Feedback Shift Register (LFSR) is an arithmetic structure which consists of a shift register and XOR (Exclusive OR) gates. The shift register is of length n and stores n bits of data at any instant. The length of the shift register is commonly referred as the length of the LFSR. The bit content stored in the shift register at a certain clock cycle is referred as the state of the LFSR within that clock cycle.[9]

Fig.3 and Fig.4 shows two possible LFSR structures, namely the Fibonacci-type and Galois-type. The LFSR memory cells are labelled with numbers 1 to 63.

As its name suggests, a LFSR operates by XORing bits inside a shift register with the bit fed to the bottommost cell. The cells in the shift register that take part in the XOR operations are commonly called the LFSR taps.

Fig.3 is an example of a 63-bit Fibonacci-type LFSR. cells 1 and 2 are chosen as taps. At each clock cycle, the bits stored in cells 1 and 2 are XORed to produce an output bit, which is simultaneously fed back to cell 63 of the shift register. All other bits in the LFSR shift one step upwards in the process. We have chosen here the output to be the XOR of cell 1 and 2.

Fig.4 is an example of a 63-bit Galois-type LFSR. cells 1 and 2 are chosen as taps. At each clock cycle, the bit stored in cell 63 is shifted to cell 1. Simultaneously the same bit is XORed with the bit originally stored in cell 1, and its outcome is sent to cell 2. All other bits in the LFSR shift one step downwards in the process. We have chosen here the output to be the bit in cell 63.

It is obvious that an LFSR is a deterministic structure. The LFSR state is continually updated at each clock cycle and will eventually it return to its original state at the beginning.

An LFSR of length n can have a maximum of $2^n - 1$ possible states. Also, for a LFSR with all bits being zero in the shift register is not considered an operational state since its output will always be zero regardless of length or combination of taps.

In general, the taps of an LFSR can be arbitrarily chosen cells. In practice we are interested in certain taps called "maximum period taps" that allow the LFSR (of fixed length) to iterate through all $2^n - 1$ possible states before repeating itself. This is to avoid repeating patterns in the LFSR output. In the examples in Fig.3 and Fig.4, cells 1 and 2 forms such a set of taps. The example LFSR will go through $2^{63} - 1$ different states before repeating outcomes start to occur.

Mathematically, an LFSR can be described by finite Galois field algebra. It can be shown that the solutions for maximimum period taps for both Fibonacci-type and Galois-type LFSRs can be analyzed by such algebra.

The output binary sequence of a LFSR with maximum period taps possess good statistical properties. The occurances of 0s and 1s are almost identical, as are the two-bit pairs (00,01,10,11) and 3-bit triplets (000,001,010...) and so on. Also, a discrete Fourier transform of the output stream shows a white spectra. Although the stream is deterministic in nature, its repetition period will be sufficiently long even for a moderate length LFSR ($2^{63} - 1$ for a 63-bit length LFSR). As such, the output stream of such a LFSR is often used as a pseudo-random number generator.

However, it should be noted that despite its good statistical properties, such a pseudo-random stream cannot be considered as good random numbers for use in information security as there exist methods to distinguish the pseudo-random stream from a LFSR from a truly random stream.[10]

The randomness extractor proposed by the applicant can be considered as a combination of a physical randomness source and a LFSR structure. The applicant proposes to utilize the good statistical properties of an LFSR output, and meanwhile also introduces the unpredictablity of a physical randomness source to construct an efficient and reliable randomness extraction scheme.

6.4 LFSR based randomness extractors

In this invention, the applicant utilizes the statistical properties of the LFSR output, but additionally seeds it with a binary sequence originating from a physical source of randomness to construct a randomness extractor. We describe first the serial version of the Fibonacci type and Galois type randomness extractors and later expand upon them to their corresponding parallel versions which are able to process all the bits in the word provided by the measurement in a single clock step.

6.4.1 Serial Fibonacci type randomness extractor

Fig.5 shows the basic structure of a serial Fibonacci type extractor. The central part is a Fibonacci type 63-bit LFSR, with the its taps chosen to give a maximal period (cells 1 and 2). The extractor has a serial input and output. The length of the LFSR is chosen to be 63 bits because it minimizes the number of tap bits (and hence XOR operations) required. In each clock cycle, the following operations take place:

1. One bit is read from the physical RNG to the input.
2. The input bit is XORed with the bits in cells 1 and 2 (labelled tap bits), generating one new bit. The operation can be expressed as: $X_{output} = X_1 \oplus X_2 \oplus X_{input}$
3. All bits in the LFSR shift one cell upwards, the vacated bottom cell is occupied by the newly generated bit. The new bit is at the same time sent to

output.

In this example, each measurement outcome from a physical randomness source is a 16-bit number which yields 12 bits of Shannon entropy. We therefore dropped 4 out of the 16 output bits to ensure an entropy of 12 bits per sample in the output stream. The LFSR is inially populated with bits from the input of the physical RNG.

In general, a longer LFSR adds more complexity to the system but also improves the statistical properties of the output. A LFSR length of < 32 bits significantly reduces the quality of the output random numbers.

The applicant notes that the structure described is very similar to "scramblers" normally used in telecommunications.[11] However, a scrambler preserves the output word length while in the randomness extractor described above one deliberately drops bits to conserve entropy.

6.4.2 Serial Galois type randomness extractor

Fig.6 shows the basic structure of a serial version, Galois-type extractor. The central part is a Galois type 63-bit LFSR, with its taps chosen to give a maximal period (cells 1 and 2). The extractor has a serial input and output. The length of the LFSR is chosen to be 63 bits because it minimizes the number of tap bits (and hence XOR operations) required. In each clock cycle, the following operations take place:

1. One bit is read from the physical RNG to the input.
2. The input bit is XORed with the output bit (bit from cell 63) of the LFSR, generating one new bit.
3. All bits in the LFSR shift one cell downwards. The vacated top cell is occupied by the newly generated bit and the bit from cell 63 is sent to the output. The XOR operation can be expressed as: $X_1 = X_{input} \oplus X_{63}, X_2 = X_1 \oplus X_{63}$

The Galois-type LFSR is mathematically equivalent to a Fibonacci-type LFSR, both structures can be described by finite Galois field algebra. The applicant notes that the serial Galois-type LFSR extractor presents a significant increase in output bit rate since all the XOR operations simultaneously occur in one clock cycle as compared to at least two clock cycles required in the serial Fibonacci-type LFSR.

6.4.3 Parallel Fibonacci type randomness extractor

Fig.7 shows the basic structure of a parallel Fibonacci-type extractor. The cells of the shift register are labelled with numbers 1 to 126. The arrows indicate the shift of bits at each clock cycle. To generate an output of 63 bits in parallel, $63 \times 2 = 126$ cells of memory are required. Cells $1 \sim 63$ store the state of the LFSR, and are to be sent to output at the next clock cycle; cells $64 \sim 126$ are used to store the next 63 bits newly generated by the XOR circuit.

Similar to the serial extractor, the operations in one clock cycle are as following:
1. 63 bits are read from the physical RNG to the input.
2. XOR operations are performed between the input 63 bits and the bits stored in the shift register following the truth table below:

XOR circuit
$X_{64} = X_1 \oplus X_2 \oplus X_{\text{input1}}$
$X_{63} = X_2 \oplus X_3 \oplus X_{\text{input2}}$
$\vdots$
$X_{126} = X_{63} \oplus X_{64} \oplus X_{\text{input63}}$

One clock cycle generates 63 new bits by performing a pair of XOR operations in sequence for every new bit.
3. All bits in the LFSR shift 63 cells forward: 64 → 1, 65 → 2, etc. The bits originally stored in cells $1 \sim 63$ are sent to output; the vacated cells $64 \sim 126$ accommodate the newly generated 63 bits from the XOR circuit.
4. A portion of bits are dropped to conserve entropy.

The applicant notes that since all the XOR operations are performed in parallel, the processing speed is 63 times faster than its serial version. This speedup is at a cost of requiring more memory space and more XOR gates. To build a parallel Fibonacci-type extractor using a n-bit LFSR that has k bit parallel output, k must be less or equal to n. The number of memory cells needed is simply $n + k$.

The entropy conservation condition and the LFSR length considerations remains the same.

6.4.4 Parallel Galois type randomness extractor

Fig.8 shows the basic structure of a parallel Galois-type extractor. The cells of the shift register are labelled with numbers 1 to 63, 64 to 126 and 64' to 126'. The arrows indicate the shift of bits at each clock cycle. To generate an output of 63 bits in parallel, $63 \times 3 = 189$ memory cells are required. Cells $1 \sim 63$ store the state of the LFSR, and are to be sent to output at next clock cycle; cells $64 \sim 189$ are used to store the 126 bits that will undergo the XOR operations.

Similar to the serial extractor, the operations in one clock cycle are as following:
1. 63 bits are read from the physical RNG to the input.
2. XOR operations are performed between the input 63 bits and the bits stored in the shift register following the truth table below:

XOR circuit 1	XOR circuit 2
$X_{64'} = X_1 \oplus X_{\text{input1}}$	$X_{64} = X_{64'} \oplus X_2$
$X_{65'} = X_2 \oplus X_{\text{input2}}$	$X_{65} = X_{65'} \oplus X_3$
$\vdots$	$\vdots$
$X_{126'} = X_{63} \oplus X_{\text{input63}}$	$X_{126} = X_{126'} \oplus X_{64}$

One clock cycle performs 126 simultaneous XOR operations and generates 63

new bits.
3. All bits in the LFSR shift 63 cells forward: $64' \to 64$, $64 \to 1$, $65' \to 65$, $65 \to 2$, etc. The bits originally stored in cells $1 \sim 63$ are sent to output; the output from XOR circuit 1 occupies cells $64' \sim 126'$; the output from XOR circuit 2 occupies cells $64 \sim 126$
4. A portion of bits are dropped to conserve entropy.

The Galois-type parallel extractor will speed up the processing speed at the cost of additional memory space and XOR operations compared to its serial version. To build a parallel Galois-type extractor using a n-bit LFSR that has a k bit parallel output, k must be less or equal to n. The number of memory cells needed would be $n + m \times k$ where m is the number of taps of the chosen LFSR.

The entropy conservation condition and the LFSR length considerations remains the same.

References

[1] Daniela Frauchiger and Renato Renner. Truly random number generation: an example. *Emerging Technologies in Security and Defence; and Quantum Security II; and Unmanned Sensor Systems X*, Oct 2013.

[2] Christian Gabriel, Christoffer Wittmann, Denis Sych, Ruifang Dong, Wolfgang Mauerer, Ulrik L. Andersen, Christoph Marquardt, and Gerd Leuchs. A generator for unique quantum random numbers based on vacuum states. *Nature Photon*, 4(10):711715, Aug 2010.

[3] P. Hellekalek. Good random number generators are (not so) easy to find. *Mathematics and Computers in Simulation*, 46(5-6):485505, Jun 1998.

[4] Benjamin Jun and Paul Kocher. The intel random number generator. 1999.

[5] Ido Kanter, Yaara Aviad, Igor Reidler, Elad Cohen, and Michael Rosenbluh. An optical ultrafast random bit generator. *Nature Photon*, 4(1):5861, Dec 2009.

[6] T. Symul, S. M. Assad, and P. K. Lam. Real time demonstration of high bitrate quantum random number generation with coherent laser light. *Applied Physics Letters*, 98(23):–, 2011.

[7] Siew-Hwee Kwok, Yen-Ling Ee, Guanhan Chew, Kanghong Zheng, Khoongming Khoo, and Chik-How Tan. A comparison of post-processing techniques for biased random number generators. *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, page 175190, 2011.

[8] John von Neumann. Various Techniques Used in Connection with Random Digits. *J. Res. Nat. Bur. Stand.*, 12:36–38, 1951.

[9] Manfred. Schroeder. *Number Theory in Science and Communication: With Applications in Cryptography, Physics, Digital Information, Computing, and Self-Similarity /; Fifth Edition.* Springer Berlin Heidelberg,, Berlin, Heidelberg :, 2009.

[10] Anne Canteaut. Berlekamp-massey algorithm. In HenkC.A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*, pages 29–30. Springer US, 2005.

[11] P. Lablans. Methods and systems for processing of n-state symbols with xor and equality binary functions, January 29 2013. US Patent 8,364,977.