LFSR based scrambler for random number processing

December 5, 2014

This LFSR(Linear Feedback Shift Register) based parallel scrambler was originally designed as a post-processing algorithm for randomness extraction. Its input receives raw random bits from a physical random number generator(RNG) and outputs a random stream that is uncorrelated and bias free. The construct is compact and can be expected to operate at ultra-fast frequency.

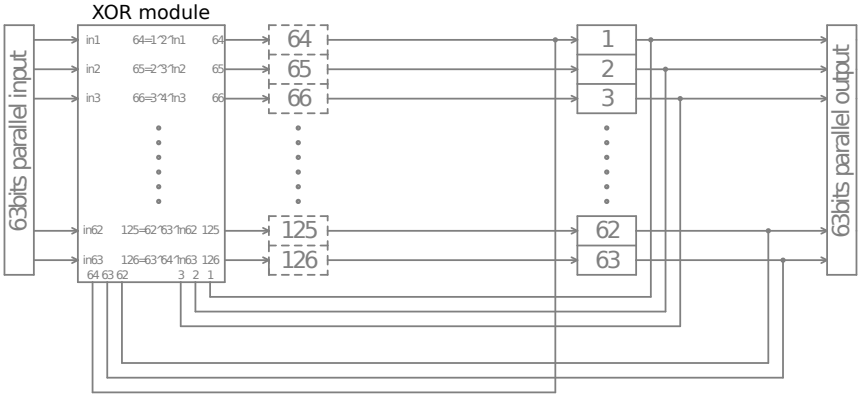

Figure 1: 63 bits parallel scrambler.

Fig 1 shows an example of a 63-bit parallel scrambler. The addresses of the shift register are labelled in numbers. 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 needed. Cells $1 \sim 63$ store the state of the LFSR, and are to be sent to output at next clock cycle; cells $64 \sim 126$ are used to store the next 63 bits that are newly generated by the XOR(Exclusive OR) module.

On each cycle, the scrambler takes in 63 bits data generated by the RNG. The bits stored in addresses $1 \sim 64$ will be read into the XOR module and XORed with the 63 input random bits. The truth table for the XOR module is shown in the figure, which can be interpreted as:

$$\begin{aligned} X_{64} &= X_1 \oplus X_2 \oplus X_{in1}, \\ X_{63} &= X_2 \oplus X_3 \oplus X_{in2}, \\ &\vdots \\ X_{126} &= X_{63} \oplus X_{64} \oplus X_{in63} \end{aligned}$$

Once the XOR operations are done, all bits in the shift register shift one step to right: bits $1 \sim 63$ go to the parallel output, bits $64 \sim 126$ occupies addresses $1 \sim 63$, and addresses $64 \sim 126$ accommodates the 63 newly generated bits from XOR module.

An attempt has been done using a 128 bit scrambler with similar structure to process the random bits generated from a quantum random number generator. Running the randomness test suite has shown no obvious bias and correlations in the output.