jet INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 2, PP. 167–171
Manuscript received January 21, 2018; revised March, 2018. DOI: 10.24425/119365

Scalable Method of Searching for Full-period Nonlinear Feedback Shift Registers with GPGPU. New List of Maximum Period NLFSRs.

Paweł Augustynowicz and Krzysztof Kanciak

***Abstract*—This paper addresses the problem of efficient searching for Nonlinear Feedback Shift Registers (NLFSRs) with a guaranteed full period. The maximum possible period for an n-bit NLFSR is $2^n - 1$ (an all-zero state is omitted). A multi-stages hybrid algorithm which utilizes Graphics Processor Units (GPU) power was developed for processing data-parallel throughput computation.**

Usage of the abovementioned algorithm allows giving an extended list of n-bit NLFSR with maximum period for 7 cryptographically applicable types of feedback functions.

***Keywords*—NLFSR, GPGPU, CUDA, Nonlinear Feedback Shift Registers, de Bruijn sequences**

I. Introduction

FEEDBACK shift registers (FSR) are used to generate binary sequences which can be applied in cryptographic systems. The first scientific studies concerning binary sequences started with the book of S.W. Golomb []. Especially stream ciphers usually use FSRs to construct invertible mappings to prevent reduction of the entropy of the internal state of a cipher.

The strongly desirable property is a long period of FSR. The period of a mapping is the length of the longest cycle in its state transition graph. For cryptographic applications, especially stream ciphers and pseudo-random numbers generators (PRNG), it is needed to prevent FSR based mapping sequence of generated states to be trapped in short cycle.

FSRs most important advantages are high speed and low power consumption. They are important primitive in lightweight cryptography [1], [2], [3]. These are crucial factors for hardware cryptographic systems especially for growing market of Internet of Things (IoT) use cases. Stream ciphers built on FSRs supports very high data rates, throughput and remains power efficient. Furthermore, there are no direct or efficient cryptoanalitic methods to easily break stream ciphers based on proper application of FSRs.

A state of a binary FSR is a vector of values of its state variables $(x_0, x_1, \ldots, x_{n-1})$. At every clock cycle, the next state of an FSR is determined from its current state by simultaneously updating the value of each stage i to the value of f_i. The period of an FSR is the length of the longest cyclic output sequence it produces. The maximum period is reached when every possible state of FSR appears exactly once. Otherwise, FSR has two or more separable cycles which period is shorter than maximal.

P. Augustynowicz and K. Kanciak are with the Faculty of Cybernetics, Military University of Technology, Warsaw, Poland (email: {krzysztof.kanciak, pawel.augustynowicz}@wat.edu.pl).

Definition 1: Formally Binary Feedback Shift Register of order n is a mapping $\mathbf{F}_2^n \rightarrow \mathbf{F}_2^n$ of the form $(x_0, x_1, ..., x_{n-1}) \rightarrow (x_1, x_2, ..., x_{n-1}, f(x_0, x_1, ..., x_{n-1}))$,

- where f is a boolean function of n variables - it is called feedback function:
  - FSR is called linear (**LFSR**) if the feedback function f is linear
  - FSR is called nonlinear (**NLFSR**) if the feedback function f is nonlinear; i.e., the function f has higher degree terms in its algebraic normal form (ANF)
- x_{n-1} is an output bit
- The period of an FSR is the length of the longest cyclic output sequence it produces.

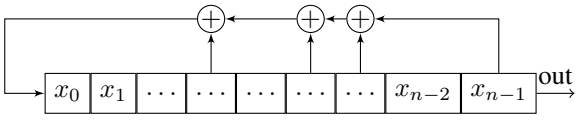

Fig. 1. A structure of Feedback Shift Register.

It is believed that the theory of construction and application of LFSRs is very mature and all crucial problems related to LFSRs have been solved. It is known that n-bit LFSR has the maximum period of $2^n - 1$ if and only if its characteristic polynomial of degree n is primitive [4]. No such rule for NLFSRs has been found so far. It has to be pointed that despite many LFSRs applications (stream ciphers, pseudo-random number generators, error detection and correction, data compression) their pseudo-random sequences are not cryptographically secure. Only $2n$ consecutive bits of LFSRs sequence is enough to deduce n-bit LFSR structure by using a Berlekamp-Massey algorithm.

Existing methods of cryptanalysis do not provide any efficient algorithm to deduce the structure of NLFSRs by only generated sequences which makes NLFSRs much more appealing for cryptographers [5]. What is more, some kinds of NLFSRs preserves all the main advantages of LFSR such as low power consumption, easy implementation and high efficiency.

In recent years, NLFSRs have received much attention in designing many cryptographic algorithms such as stream ciphers (for example GRAIN which is NIST standard [6], Trivium [7] or Achterbahn [8]), lightweight block ciphers and sponge-based generators [1] like KATAN/KTANTAN [9]. In most cases, NLFSRs have much greater linear complexity than LFSRs of the same period, which is directly connected with the security of cryptographic algorithms [10].

Sequences generated by full period NLFSRs are also known as de Bruijn sequences. In a de Bruijn sequence of order n all 2^n different binary n-tuples appear exactly once. It was proved by Flye Sainte-Marie in 1894 and independently by de Bruijn in 1946 that the number of cyclically inequivalent sequences is equal to $2^{2^{n-1}-n}$. Therefore, the number of potential NLFSR's feedback functions is much greater than the number of LFSR's feedback functions of the same degree which is equal $\frac{\phi(2^n-1)}{n}$.

Not all possible NLFSRs are directly applicable to cryptographic applications. There are several properties that must be fulfilled by these cryptographic primitives, particularly:

- the number of feedback function's nonlinear terms should remain as small as possible;
- the algebraic degree of feedback function should be at most 3,
- the number of linear terms in feedback function should not exceed 6 due to lightweight cryptography requirements.

II. Generation of NLFSRs

An efficient method of construction cryptographically strong and efficient NLFSRs remains unknown. The most important NLFSR related problem is finding a systematic procedure for constructing NLFSRs with a guaranteed long period. Available algorithms either consider some special cases, or are applicable to small NLFSRs only, or produce NLFSRs which are not cryptographically applicable [11], [12], [13].

A. Previous methods of generation

There are at least several different approaches to generating NLFSRs with desired cryptographic properties. The most common strategy of searching is exhaustive testing of potential nonlinear feedback functions. That strategy was employed by Janusz Szmidt [13]. He has tested potential feedback functions with a simple algebraic normal form for a maximum period on 54 CPU cores, which allows to testing 54 potential feedback functions at once. As it turned out the number of cores is insufficient for effective search due to the size of the search space. The number of polynomials to be tested growths rapidly with the polynomial degree. For example, for the simplest form of NFLSR: $x_i \oplus x_j \oplus x_k \oplus x_l \cdot x_m$ there are $\frac{1}{6}n^2 \cdot (n-1)^2 \cdot (n-2)$ possible feedback polynomials of degree n (see Figure 2).

Another approach is to develop an special purpose hardware implemented on Field Programmable Gate Arrays (FPGA) [14][13]. FPGAs circuits allow developing highly efficient and parallel search algorithms. As it is mentioned in [14], using FPGAs special hardware it is possible to obtain at least 150 speed-up comparing to computing power of standard Intel

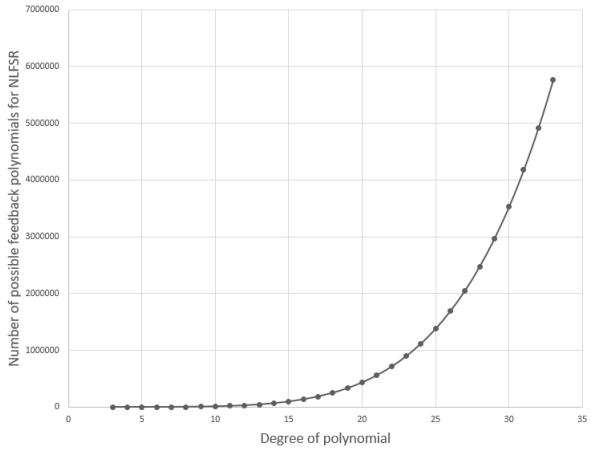

Fig. 2. Growth of the number of possible feedback functions with the degree.

Core i5-3210M CPU 2,5 GHz processor and make it possible to test about 150 NLFSRs with cryptographic applicable form simultaneously. The above-mentioned approach results in generating some NLFSR with special structure up to degree 29.

The above-mentioned results, in comparison with the number of all possible feedback functions for the NLFSR ($2^{2^{n-1}-n}$), where n is the degree of the polynomial [15]), are insufficient. The rapid growth of the number of possible feedback functions implies that thousands and millions of threads should be used to test possible polynomials efficiently and find at least few applicable ones. That was the main reason for applying General Purpose Graphics Processor Units (GPGPU) in the development of our search method.

GPGPUs are valued for their efficiency, high throughput and a possibility of using thousands of threads at once. They were designed to perform simple, independent task in parallel and efficient manner. That was the underlying reason for applying them in our search for NLFSRs. Furthermore, the peak efficiency of modern GPU is growing increasingly and that growth is much faster than for CPU (see figure 3).

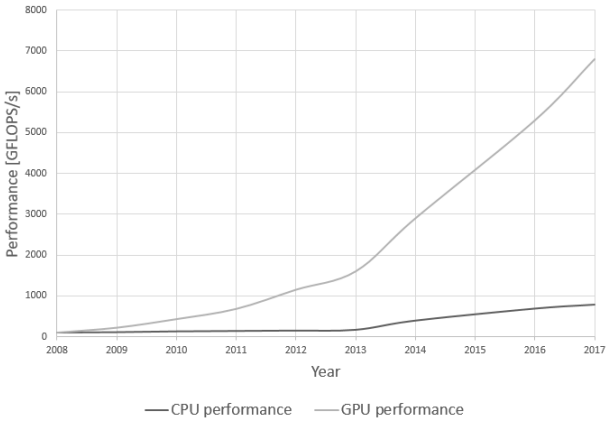

Fig. 3. Comparison of CPU and GPU performance over past 10 years.

B. Outline of our method

The main assumption of our methodology was the fact that modern GPUs like NVIDIA GeForce GTX 1080 are equipped with up to 2560 cores. Furthermore, the frequency of cores, especially in boost mode is very high. It results in very good general performance of GPUs in comparison to CPUs and most of FPGAs.

Our method like other methods presented [11], [14], [13] consists of two main phases:

1) Generation of possible feedback polynomials.
2) Verification of their period.

For the purpose of systematic search the generation of possible feedback functions is conducted on CPU and the memory is asynchronously transferred to the global memory of GPU. If we are interested not in the systematic search, but in finding any feedback polynomial of given degree, generation phase could be easily performed on GPU with a usage of a random number generation library, for example, cuRAND [16]. The global memory of GPU can be very large in modern graphics cards but its main drawback is the long average access time. Every possible function is represented by two integer variables: one for linear part of the function and the second one for nonlinear. Then every thread is reading his own feedback function and stores it in its local memory that has very short access time. An additional local variable is created to maintain the state of the NLFSR.

The first stage of designed process gathers a set of polynomials to be tested. The whole process is designed to make testing polynomials (for being maximal period NLFSR) so-called embarrassingly parallel task which means it is easy to divide a huge set of suspected polynomials into chunks that can be processed on single GPU separately.

Next step is verification of the period of candidate feedback function which is performed in a simple loop. After the $2^n - 1$ iteration of the loop, the state of the register is checked. We do not store any intermediate register states for this moment (low memory complexity). If the starting and ending point are the same the function is expected to be the feedback function of maximum period NLFSR. Therefore it is transferred to CPU for an additional test in which we check if register generates full cycle with all possible states. This stage works like a sieve which rejects the vast majority of suspected polynomials which give shorter than full periods and runs entirely on GPUs. This stage is computationally hard, but designed to run in parallel (no 'if' statements). It is worth mentioning that optimization techniques were applied like CUDA integer intrinsics utilization ($\_\_popc$ and $\_\_popcll$ functions for counting the number of bits that are set to 1 in integer) and using decimal arithmetic when creating a new state of the register. GPU task granularity depends on a degree of tested polynomials, but for degrees bigger than 20 the granulation is big enough to hide overhead of copying memory between host and GPU device.

The last stage of the process confirms if polynomials tested in the second stage give full period. In trivial approach, last stage stores all intermediate register states, but some optimization techniques were applied to relax the memory complexity of this stage. It runs on CPUs, but very few of polynomials have to be fully checked during the last stage of the process. In the worst case, CPU stage takes about 10 times less than the GPU stage, therefore this computations have no effect on the efficiency of GPU search. All mentioned above make the algorithm data-parallel that scales out very well in multicore heterogenous architecture.

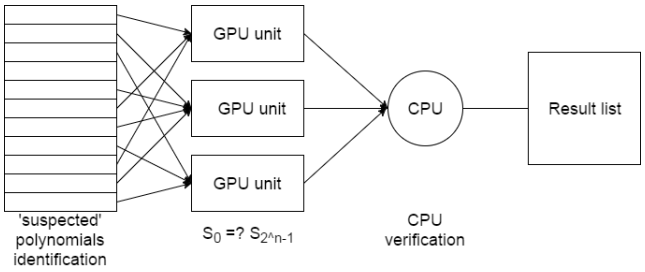

Fig. 4. Presentation of GPU verification method.

The above-mentioned approach has several advantages over other methods:

- it is easily scalable,
- the same code can be applied on any GPU despite the model,
- it can be used to search for any kind of linear or nonlinear feedback function (FPGA methods are developed for a fixed form of feedback function),
- in this particular task GPU has better efficiency than CPU-based or FPGA-based solutions.

III. GPU-BASED NLFSR PERFORMANCE ANALYSIS

In [14] hardware performance was defined as a number of tested NLFSRs per time unit, denoted as $\#NLFSR$. Another hardware performance indicator discussed here is the time of one cycle (denoted as t_{cycle}^{GPU}). The cycle is defined as a going through the all possible states of NLFSR and returning to initial state. The third vital characteristic mentioned in [14] (the number of simultaneously tested NLFSRs per cycle) for GPUs is constant and dependent on the type of used graphic card. Our method was applied on NVIDIA GeForce 1080GTX which is equipped with 2560 CUDA cores.

Comparing our method to the previous ones it can be seen that GPU computation method is the most efficient and scalable one. For example, on GPU we are able to test on average 311 NLFSRs of degree 29 on time unit whereas on FPGA the result is about two times worse [14].

In conclusion, our GPU method of searching NLFSR is the most efficient and scalable one. Nevertheless, from the cryptographic point of view, the NLFSR with degree 30 is not sufficient for cryptographic applications due to security reasons [17]. Therefore, the efficiency and applicability of searching methods should still be developed and improved.

TABLE I
NUMBER OF TESTED NLFSR PER TIME UNIT AND TIME OF ONE CYCLE FOR NVIDIA GEFORCE 1080GTX.

degree	$\#NLFSR$	t_{cycle}^{GPU}
20	233524	$1,67 \cdot 10^{-9}$
21	122597	$3,18 \cdot 10^{-9}$
22	63226	$6,17 \cdot 10^{-9}$
23	32874	$1,19 \cdot 10^{-8}$
24	16590	$2,35 \cdot 10^{-8}$
25	8207	$4,76 \cdot 10^{-8}$
26	4176	$9,35 \cdot 10^{-8}$
27	1763	$2,21 \cdot 10^{-7}$
28	877	$4,40 \cdot 10^{-7}$
29	311	$1,19 \cdot 10^{-6}$
30	170	$2,30 \cdot 10^{-6}$

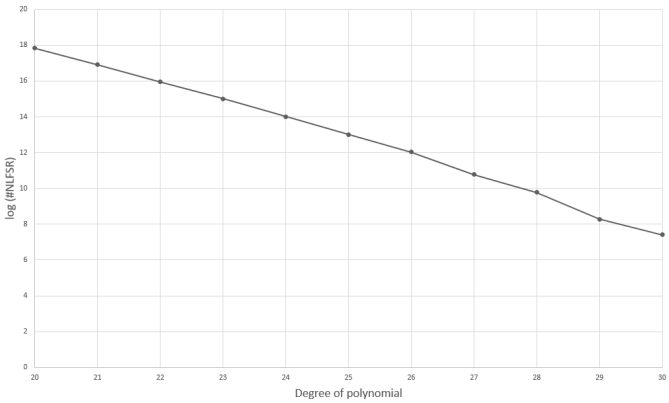

Fig. 5. The decrease of number of tested NLFSRs per time unit with the degree of NLFSR.

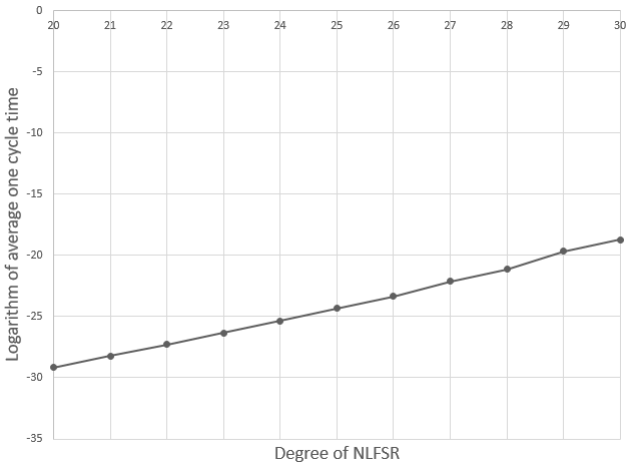

Fig. 6. The increase of average time of one cycle with the degree of NLFSR.

TABLE II
NUMBER OF FOUND NLFSR OF GIVEN DEGREE AND FORM.

form	12	14	16	18	20	22	24	26
(3:2)	3	1	1	0	0	0	0	0
(3:3)	17	11	3	1	2	1	0	0
(5:2)	36	28	16	10	4	6	2	0
(5:3)	58	58	104	20	8	0	0	0
(2:2:2)	30	20	12	4	0	4	0	0
(4:2:2)	1368	1116	812	512	252	172	60	28
(4:3:2)	26	2	2	2	0	0	0	0

IV. SEARCH FOR NLFSRS OF SPECIAL FORMS

In [18] only some special forms of NLFSR were searched. They all characterized a small number of terms and very elegant form. As a result, it was possible to find all maximum period NLFSRs of that form due to reasonable for CPU search space. We have applied the similar methodology of searching for NLFSR. Usage of GPU allows expanding reasonable search space for new forms of possible feedback functions. As a result of our research following types of feedback functions polynomials were tested:

- $x_a \oplus x_b \oplus x_c \oplus x_d \cdot x_e$: (3:2),
- $x_a \oplus x_b \oplus x_c \oplus x_d \cdot x_e \cdot x_f$: (3:3),
- $x_a \oplus x_b \oplus x_c \oplus x_d \oplus x_e \oplus x_f \cdot x_g$: (5:2),
- $x_a \oplus x_b \oplus x_c \oplus x_d \oplus x_e \oplus x_f \cdot x_g \cdot x_h$: (5:3),
- $x_a \oplus x_b \oplus x_c \cdot x_d \oplus x_e \cdot x_f$: (2:2:2),
- $x_a \oplus x_b \oplus x_c \oplus x_d \oplus x_e \cdot x_f \oplus x_g \cdot x_h$: (4:2:2),
- $x_a \oplus x_b \oplus x_c \oplus x_d \oplus x_e \cdot x_f \cdot x_g \oplus x_h \cdot x_i$: (4:3:2),
- $x_a \oplus x_b \oplus x_c \oplus x_d \oplus x_e \oplus x_f \oplus x_g \cdot x_h \oplus x_i \cdot x_j$: (6:2:2).

Numbers of NLFSRs with polynomial of given structure are presented in the table below.

As it can be seen from table II the number of the NLFSRs with given form has a strong tendency to decrease with the degree. Furthermore, it can be seen that in general there is a correlation between the number of terms of polynomial and the number of found NLFSR of given degree. What is more, it can be seen that there are no full period NLFSR with high degree and very nice form like (3:2) or (5:2). It can be concluded that searching for maximum period NLFSR will always be a trade-off between its form applicability and length of the period.

From the cryptographic point of view, the forms $x_a \oplus x_b \oplus x_c \oplus x_d \oplus x_e \oplus x_f \oplus x_g \cdot x_h \oplus x_i \cdot x_j$ (6:2:2) and $x_a \oplus x_b \oplus x_c \oplus x_d \oplus x_e \cdot x_f \oplus x_g \cdot x_h$ (4:2:2) are the most appealing ones. They can be implemented in FPGA in a efficient manner and we found all feedback functions of this form up to degree 28.

Examples of NLFSRs of structure (6:2:2) with degrees 25, 26 and 28 can be found below:

(6:2:2) — degree 25

$x_{24} \oplus x_{22} \oplus x_8 \oplus x_2 \oplus x_1 \oplus x_0 \oplus x_4 \cdot x_9 \oplus x_4 \cdot x_{13}$
$x_{24} \oplus x_{15} \oplus x_{12} \oplus x_2 \oplus x_1 \oplus x_0 \oplus x_5 \cdot x_{22} \oplus x_{12} \cdot x_{16}$
$x_{24} \oplus x_{14} \oplus x_4 \oplus x_3 \oplus x_1 \oplus x_0 \oplus x_2 \cdot x_0 \oplus x_9 \cdot x_8$
$x_{24} \oplus x_{22} \oplus x_6 \oplus x_3 \oplus x_1 \oplus x_0 \oplus x_{12} \cdot x_0 \oplus x_{13} \cdot x_6$
$x_{24} \oplus x_{12} \oplus x_6 \oplus x_4 \oplus x_1 \oplus x_0 \oplus x_{12} \cdot x_4 \oplus x_{16} \cdot x_4$
$x_{24} \oplus x_{18} \oplus x_{11} \oplus x_4 \oplus x_1 \oplus x_0 \oplus x_{15} \cdot x_7 \oplus x_{16} \cdot x_8$
$x_{24} \oplus x_{23} \oplus x_{21} \oplus x_4 \oplus x_1 \oplus x_0 \oplus x_{19} \cdot x_{13} \oplus x_{22} \cdot x_{15}$
$x_{24} \oplus x_{10} \oplus x_7 \oplus x_5 \oplus x_1 \oplus x_0 \oplus x_5 \cdot x_4 \oplus x_{17} \cdot x_7$
$x_{24} \oplus x_{19} \oplus x_7 \oplus x_5 \oplus x_1 \oplus x_0 \oplus x_{14} \cdot x_5 \oplus x_{21} \cdot x_{19}$
$x_{24} \oplus x_{16} \oplus x_{14} \oplus x_5 \oplus x_1 \oplus x_0 \oplus x_{14} \cdot x_5 \oplus x_{16} \cdot x_{12}$

(6:2:2) — degree 26

$x_{25} \oplus x_{22} \oplus x_{11} \oplus x_3 \oplus x_1 \oplus x_0 \oplus x_{11} \cdot x_0 \oplus x_{21} \cdot x_{14}$
$x_{25} \oplus x_{20} \oplus x_{14} \oplus x_3 \oplus x_1 \oplus x_0 \oplus x_{24} \cdot x_1 \oplus x_{18} \cdot x_{17}$
$x_{25} \oplus x_{11} \oplus x_5 \oplus x_4 \oplus x_1 \oplus x_0 \oplus x_7 \cdot x_2 \oplus x_{16} \cdot x_{10}$

$$x_{25} \oplus x_{13} \oplus x_7 \oplus x_4 \oplus x_1 \oplus x_0 \oplus x_9 \cdot x_3 \oplus x_{23} \cdot x_{11}$$
$$x_{25} \oplus x_{12} \oplus x_8 \oplus x_4 \oplus x_1 \oplus x_0 \oplus x_8 \cdot x_{20} \oplus x_{17} \cdot x_{23}$$
$$x_{25} \oplus x_{17} \oplus x_{13} \oplus x_4 \oplus x_1 \oplus x_0 \oplus x_{15} \cdot x_3 \oplus x_{18} \cdot x_8$$
$$x_{25} \oplus x_{15} \oplus x_{13} \oplus x_5 \oplus x_1 \oplus x_0 \oplus x_{15} \cdot x_7 \oplus x_{24} \cdot x_{21}$$
$$x_{25} \oplus x_{21} \oplus x_{13} \oplus x_5 \oplus x_1 \oplus x_0 \oplus x_{15} \cdot x_{10} \oplus x_{19} \cdot x_{24}$$
$$x_{25} \oplus x_{22} \oplus x_{10} \oplus x_6 \oplus x_1 \oplus x_0 \oplus x_{24} \cdot x_4 \oplus x_{13} \cdot x_7$$
$$x_{25} \oplus x_{18} \oplus x_{17} \oplus x_9 \oplus x_1 \oplus x_0 \oplus x_0 \cdot x_{17} \oplus x_{16} \cdot x_3$$

(6:2:2) — degree 28

$$x_{27} \oplus x_{24} \oplus x_{14} \oplus x_2 \oplus x_1 \oplus x_0 \oplus x_{12} \cdot x_4 \oplus x_4 \cdot x_{16}$$
$$x_{27} \oplus x_{22} \oplus x_{13} \oplus x_5 \oplus x_1 \oplus x_0 \oplus x_1 \cdot x_1 \oplus x_{17} \cdot x_5$$
$$x_{27} \oplus x_{20} \oplus x_{14} \oplus x_5 \oplus x_1 \oplus x_0 \oplus x_1 \cdot x_0 \oplus x_5 \cdot x_{17}$$
$$x_{27} \oplus x_{20} \oplus x_{18} \oplus x_6 \oplus x_1 \oplus x_0 \oplus x_{24} \cdot x_8 \oplus x_{25} \cdot x_{21}$$
$$x_{27} \oplus x_{17} \oplus x_9 \oplus x_8 \oplus x_1 \oplus x_0 \oplus x_7 \cdot x_2 \oplus x_{11} \cdot x_{25}$$
$$x_{27} \oplus x_{12} \oplus x_{10} \oplus x_8 \oplus x_1 \oplus x_0 \oplus x_{25} \cdot x_0 \oplus x_{17} \cdot x_{24}$$
$$x_{27} \oplus x_{24} \oplus x_{22} \oplus x_{10} \oplus x_1 \oplus x_0 \oplus x_{19} \cdot x_4 \oplus x_{12} \cdot x_{20}$$
$$x_{27} \oplus x_{22} \oplus x_{15} \oplus x_{11} \oplus x_1 \oplus x_0 \oplus x_6 \cdot x_2 \oplus x_{10} \cdot x_{20}$$
$$x_{27} \oplus x_{21} \oplus x_{16} \oplus x_{10} \oplus x_1 \oplus x_0 \oplus x_{14} \cdot x_4 \oplus x_{16} \cdot x_{24}$$
$$x_{27} \oplus x_{25} \oplus x_{21} \oplus x_{16} \oplus x_1 \oplus x_0 \oplus x_{10} \cdot x_{24} \oplus x_{22} \cdot x_{25}$$

Examples of NLFSRs of structure (4:2:2) with degrees 25, 26, 27 and 28 are presented below:

(4:2:2) — degree 25

$$x_{24} \oplus x_9 \oplus x_2 \oplus x_1 \oplus x_{12} \cdot x_8 \oplus x_{17} \cdot x_{14}$$
$$x_{24} \oplus x_{10} \oplus x_3 \oplus x_1 \oplus x_9 \cdot x_{21} \oplus x_{10} \cdot x_{16}$$
$$x_{24} \oplus x_{13} \oplus x_4 \oplus x_1 \oplus x_{13} \cdot x_8 \oplus x_{14} \cdot x_{21}$$
$$x_{24} \oplus x_{12} \oplus x_7 \oplus x_1 \oplus x_{10} \cdot x_{20} \oplus x_{15} \cdot x_{18}$$
$$x_{24} \oplus x_{19} \oplus x_{18} \oplus x_1 \oplus x_{10} \cdot x_8 \oplus x_{17} \cdot x_{16}$$

(4:2:2) — degree 26

$$x_{25} \oplus x_{17} \oplus x_3 \oplus x_2 \oplus x_1 \cdot x_5 \oplus x_4 \cdot x_{21}$$
$$x_{25} \oplus x_{24} \oplus x_3 \oplus x_2 \oplus x_2 \cdot x_{12} \oplus x_{14} \cdot x_{24}$$
$$x_{25} \oplus x_7 \oplus x_4 \oplus x_2 \oplus x_6 \cdot x_4 \oplus x_{14} \cdot x_{12}$$
$$x_{25} \oplus x_{11} \oplus x_{10} \oplus x_4 \oplus x_3 \cdot x_{14} \oplus x_9 \cdot x_8$$
$$x_{25} \oplus x_{17} \oplus x_6 \oplus x_5 \oplus x_8 \cdot x_{17} \oplus x_{17} \cdot x_{18}$$

(4:2:2) — degree 27

$$x_{26} \oplus x_{16} \oplus x_{10} \oplus x_1 \oplus x_6 \cdot x_{14} \oplus x_6 \cdot x_{21}$$
$$x_{26} \oplus x_{16} \oplus x_5 \oplus x_3 \oplus x_2 \cdot x_0 \oplus x_1 \cdot x_{21}$$
$$x_{26} \oplus x_{22} \oplus x_{18} \oplus x_2 \oplus x_1 \cdot x_{10} \oplus x_4 \cdot x_{12}$$
$$x_{26} \oplus x_{15} \oplus x_6 \oplus x_3 \oplus x_{10} \cdot x_{12} \oplus x_{22} \cdot x_{23}$$
$$x_{26} \oplus x_{13} \oplus x_9 \oplus x_5 \oplus x_8 \cdot x_{15} \oplus x_{19} \cdot x_{17}$$

(4:2:2) — degree 28

$$x_{27} \oplus x_{22} \oplus x_3 \oplus x_1 \oplus x_3 \cdot x_4 \oplus x_7 \cdot x_{21}$$
$$x_{27} \oplus x_{25} \oplus x_{23} \oplus x_4 \oplus x_5 \cdot x_{19} \oplus x_{23} \cdot x_{22}$$
$$x_{27} \oplus x_{19} \oplus x_{10} \oplus x_5 \oplus x_0 \cdot x_{19} \oplus x_5 \cdot x_{14}$$
$$x_{27} \oplus x_{15} \oplus x_6 \oplus x_3 \oplus x_{10} \cdot x_{12} \oplus x_{22} \cdot x_{23}$$
$$x_{27} \oplus x_{26} \oplus x_{17} \oplus x_{15} \oplus x_3 \cdot x_{20} \oplus x_3 \cdot x_{24}$$

V. Conclusion

In the paper, we have proposed a scalable method of searching for NLFSRs that runs well in multicore heterogenous architecture. It achieves better performance results than previously known approaches. More precisely method of efficient period checking has been developed. Many tests have been performed for chosen polynomials tests. Ongoing work is focused on methods of identifying polynomials and its forms that give better chance for a full period generation.

We performed the full search for NLFSRs of special forms. We have found NLFSRs that make up the list of known full-period NLFSRs. However, our list of full-period NLFSRs is still not sufficient for cryptographic applications [17], [10] and the search should be continued for higher degrees.

References

[1] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark: A lightweight hash," in *Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems*, ser. CHES'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1–15. [Online]. Available: http://dl.acm.org/citation.cfm?id=1881511.1881513

[2] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *Int. J. Wire. Mob. Comput.*, vol. 2, no. 1, pp. 86–93, may 2007. [Online]. Available: http://dx.doi.org/10.1504/IJWMC.2007.013798

[3] H. H. Vahid Amin Ghafari and Y. Chen, "Fruit-v2: Ultra-lightweight stream cipher with shorter internal state," 2016. [Online]. Available: http://eprint.iacr.org/2016/355

[4] S. Golomb, *Shift Register Sequences*, 1967, portions co-authored with Lloyd R. Welch, Richard M. Goldstein, and Alfred W. Hales.

[5] A. Canteaut, "Open problems related to algebraic attacks on stream ciphers," in *Proceedings of the 2005 International Conference on Coding and Cryptography*, ser. WCC'05. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 120–134. [Online]. Available: http://dx.doi.org/10.1007/11779360_10

[6] H. Zhang and X. Wang, "Cryptanalysis of stream cipher Grain family," 2009. [Online]. Available: http://eprint.iacr.org/2009/109

[7] C. D. Canniere and B. Preneel, "Trivium specifications," *eSTREAM, ECRYPT Stream Cipher Project*, vol. 2006.

[8] B. M. Gammel, R. Gottfert, and O. Kniffler, "The achterbahn stream cipher," 2005.

[9] C. De Cannière, O. Dunkelman, and M. Knežević, *KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers*. Springer Berlin, 2009.

[10] N. T. Courtois and W. Meier, *Algebraic Attacks on Stream Ciphers with Linear Feedback*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 345–359.

[11] P. Dabrowski, G. Labuzek, T. Rachwalik, and J. Szmidt, "Searching for nonlinear feedback shift registers with parallel computing," 2013.

[12] E. Dubrova, "A scalable method for constructing galois NLFSRs with period $2^n - 1$ using cross-join pairs," 2011. [Online]. Available: http://eprint.iacr.org/2011/632

[13] T. Rachwalik, J. Szmidt, R. Wicik, and J. Zablocki, "Generation of nonlinear feedback shift registers with special-purpose hardware," 2012. [Online]. Available: http://eprint.iacr.org/2012/314

[14] N. Poluyanenko, "Development of the search method for non-linear shift registers using hardware implemented on field programmable gate arrays," 2017.

[15] E. Dubrova, "Generation of full cycles by a composition of nlfsrs," *Des. Codes Cryptography*, vol. 73, no. 2, pp. 469–486, nov 2014. [Online]. Available: http://dx.doi.org/10.1007/s10623-014-9947-3

[16] *CURAND Library: Programming Guide, Version 7.0*, NVIDIA, 2015. [Online]. Available: http://docs.nvidia.com/cuda/curand

[17] M. Afzal and A. Masood, "Algebraic cryptanalysis of a nlfsr based stream cipher," in *2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications*, 4 2008, pp. 1–6.

[18] E. Dubrova, "A list of maximum period NLFSRs," 2012.