# 1.1. ОСНОВНИ МЕТОДИ ЗА ГЕНЕРАЦИЯ НА СЛУЧАЙНИ ПОСЛЕДОВАТЕЛНОСТИ

Прието е събитие, чието настъпване не може да се предскаже с точност и зависи от сложен и неизвестен по своята природа комплекс условия, да се нарича случайно. Подобно събитие се отъждествява със случайна величина, чиято конкретна стойност не може да се определи предварително [7, 13].

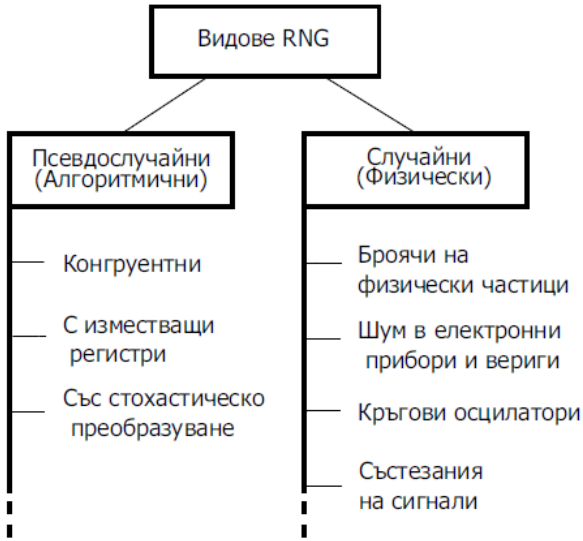
Във връзка с автоматичната генерация на случайни числа, Кнут обръща внимание на това, че едно отделно взето число не може да се определи като случайно [34]. Това свойство се проявява единствено в достатъчно голяма последователност от стойности на величината. Подобна последователност се нарича *случайна последователност*. Случайната последователност може да се разглежда като времеви ред от независими стойности с характерен закон за разпределение на плътността на вероятността, т.е като дискретен случаен процес [7, 11].

Случайните последователности имат широко приложение в моделирането, числените методи, приемането на решения, криптографията, кодовете за намиране и отстраняване на грешки и др. [1, 4, 6, 8, 34, 39, 44, 47]. Изработват се от специализирани компоненти, наричани *генератори на случайни последователности*, заменили използваните преди това за целта таблици.

Под *генератор на случайни числа* (*Random Number Generator*, *RNG*) се разбира устройство или алгоритъм, което изработва на изхода си независими случайни стойности [39].

Един от първите апаратни генератори на случайни последователности е реализиран в компютъра *Ferranti Mark I*. Този компютър притежавал специална машинна инструкция за изработване на *20* случайни бита. За ентропиен източник по препоръка на Тюринг бил използван резистивен шум.

Генераторите на случайни последователности се делят на два основни вида – псевдослучайни и случайни, както е показано на фиг. 1.1. *Псевдослучайните генератори* (Pseudo-Random Generators, *PRNG*) в основата си са алгоритмически, докато случайните или *действително-случайните* (Real Random Generators, *RRNG*; True Random Generators, *TRNG*) – физически.




**Фиг. 1.1. Класификация на основните видове *RNG***

Един от най-популярните начини, използван при *PRNG* за изработване на псевдослучайни последователности, се основава на метода за генерация на линейни конгруентни[1] последователности. Предложен е през 1951 от Lehmer [6, 34]. При зададена начална стойност $X_0$, елементите $X_{i+1}$ на последователността се определят от рекурентната зависимост

$$X_{i+1} = (X_i a + b) \bmod m \qquad (1.1)$$

При подходящи параметри *a*, *b* и *m*, генерираната последователност проявява добри случайни свойства, но е периодична с максимален период *m*. Поради периодичността на генерираната последователност, тя се разглежда като *псевдослучайна*. Максималният период *m* се постига, тогава и само тогава, когато са изпълнени условията:

- стъпката *b* и модулът *m* са взаимнопрости;
- за всяко просто число *p*, ако *m* е кратно на *p*, то и (*a - 1*) е кратно на *p*;
- ако *m* е кратно на *4*, то и (*a - 1*) е кратно на *4*.

Конкретни стойности на тези параметри са посочени в таблица 16.1 на [47].

Генераторите, използващи зависимостта (1.1), съответно се наричат *линейни конгруенти генератори* (*LCG*). Библиотечните функции

1 *Конгруетност* - отношение на еквивалентност, тъждественост. Например две фигури са конгруентни, ако съществува такова преместване, което привежда едната от тях в другата [7].

*rand()* и *srand()* на *C* и *C++* са именно от типа *LCG*, но конкретните стойности на параметрите *a*, *b* и *m* не са стандартизирани. Например в стандарта *ISO-C* [3, 32] е приведена следната препоръчителна реализация на функцията *srand()*

```
static unsigned long int next = 1;

int rand(void) // RAND_MAX assumed to be 32767
{
   next = next * 1103515245 + 12345;
   return (unsigned int)(next/65536) % 32768;
}

void srand(unsigned int seed)
{
   next = seed;
}
```

Променливата на състоянието *next* е с начална стойност *1*. Тя може да се променя чрез обръщение към рандомизиращата функция *srand()*. Функцията *rand()* изработва псевдослучайна последователност с равномерно разпределение в диапазона от *0* до *32767*.

В литературата не се препоръчва използването на линейни (*LCG*) и мултипликативни (*MLCG*) генератори освен в отделни редки случаи [34, 44]. Като следствие, това стеснява и кръга на приложение на библиотечните реализации *rand()* и *srand()* при *C* и *C++*. За преодоляване на това ограничение, в последната версия на стандарта *C++11* на езика *C++* са предвидени множество машини (*random engines*[2]), като източници на ентропия. Въпреки това, в [33] се отбелязва: „нищо в компютъра не е действително случайно, откъдето следват значителните усилия, необходими за да се осигури достатъчна степен на случайност“[3].

Генераторите на псевдослучайни последователности с *изместващи регистри с линейна обратна връзка* (Linear Feedback Shift Registers, *LFSR*) са важен клас. Основават се на теорията на линейните последователностни автомати и на теорията на крайните полета на Галоа. Основните им предимства са: ефективната апаратна и програмна реализация, добрите статистически свойства на генерираните последователности, възможността да се използват

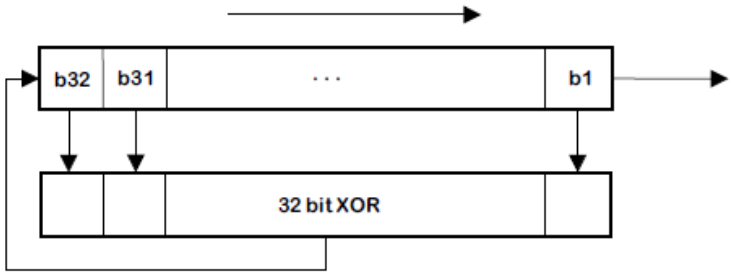
2 Достъпни чрез заглавния файл <*random*>.

3 Твърдението, че в компютъра няма източници на действително случаен сигнал, разбира се е невярно. В случая се има предвид, че сигналът, изработван от тези източници, не е в удобен за използване вид.

като компоненти на по-сложни генератори.

Примери за приложения на *RNG* с *LFSR* са формирането на *CRC* код, поточните шифри *A5*, *PANAMA*, *SOBER*, блоковият шифър *RIJNDAEL*.

*LFSR* генераторите се състоят от два основни блока – изместващ регистър и верига за обратна връзка (фиг. 1.2). Изходът на генератора е последователен и отговаря на младшия разряд $b_1$ на изместващия регистър. Вариантът от фиг. 1.2 се нарича *конфигурация на Фибоначи*.

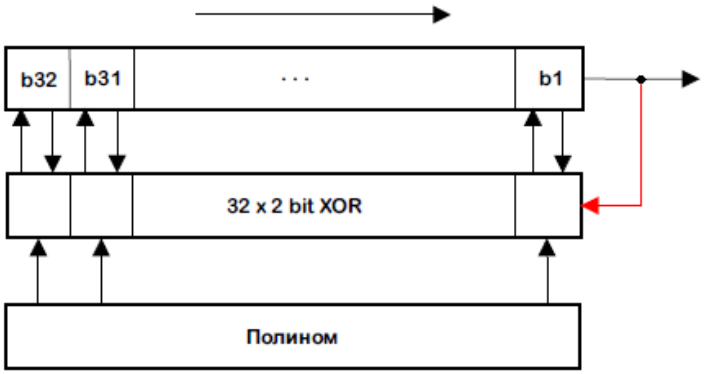



**Фиг. 1.2. *LFSR* генератор – конфигурация Фибоначи**

Функцията във веригата за обратна връзка е линейна и представлява *изключващо ИЛИ* (*XOR*) на съдържанието на определени битове от регистъра. Тези битове се определят от образуващия полином. Степента на полинома се равнява на разрядността на изместващия регистър.

Подобно на *LCG* и *LFSR* генерират периодична последователност. Интерес са *LFSR* с максимален период, които използват примитивни образуващи полиноми. Някои примитивни полиноми са приведени в таблица 16.2 на [47].

Схемата на обратната връзка от фиг. 1.2 е възможно да се модифицира във вида, посочен на фиг. 1.3. Докато в първия случай се



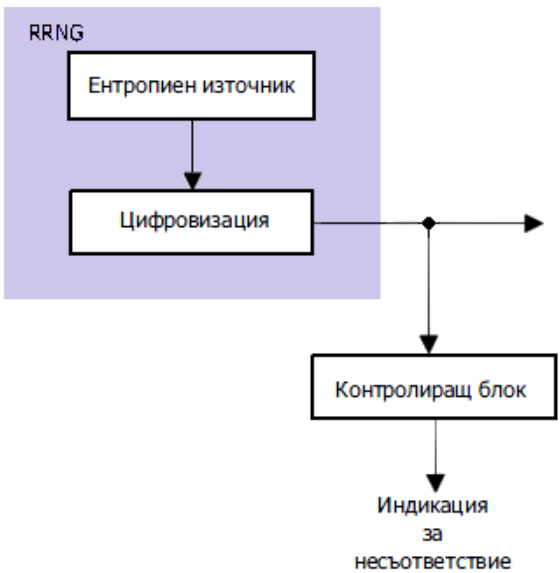

**Фиг. 1.3. *LFSR* генератор – конфигурация Галоа**

използва един *32*-входен *XOR* елемент, тук се използват *32* двувходови *XOR* елемента. Подобна схема е известна като *конфигурация на Галоа*. Апаратната сложност и на двете конфигурации е еднаква, но софтуерната реализация на конфигурацията на Галоа е много по-ефективна – една единствена побитова операция *XOR* е достатъчна да се формира бит $b_{32}$ на изхода на веригата за обратна връзка.

Генераторите на псевдослучайни последователности с изместващи регистри с обратна връзка са в основата на потоковите шифри. За увеличаване на сигурността им, т.е. за подобряване на статистическите им качества, те се комбинират. Някои от множеството методи за комбиниране на *LFSR* генераторите в по-сложни са разгледани в [47].

Кнут предлага алгоритъм за комбиниране на две псевдослучайни последователности и го обозначава като *Алгоритъм-М* [6, 34, 47]. Алгоритъмът използва стохастическо преобразуване и е независим от вида на генераторите, изработващи двете последователности. Разглежда се по-детайлно в т. 2.1.

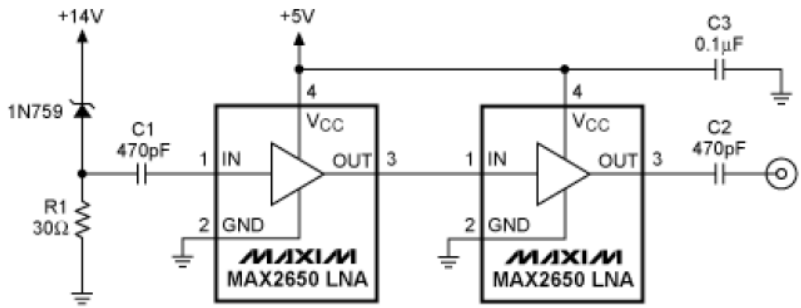
Вторият основен вид генератори от фиг. 1.1 изработват действително-случайни последователности и се наричат съответно генератори на *действително-случайни* последователности (Real Random Generators, *RRNG* или True Random Generators, *TRNG*) или просто – генератори на случайни последователности. В основата им са природни явления, откъдето идва и другото название на тези генератори – *физически*.




**Фиг. 1.4. Блокова схема на физически генератор (*RRNG*)**

Общата блокова схема на физическите генератори е представена на фиг. 1.4. Структурата на *RRNG* съдържа два основни блока – блока за смъкване на сигнал от ентропийния източник и цифровизиращия блок. За ентропиен източник се използват различни природни явления – естествения радиоактивен фон, шумът в електронни прибори и вериги и др. Изискването към ентропийния източник е да осигурява сигнал, близък до белия шум, чиято плътност на спектъра на мощността е константа [11].

Контролиращият блок следи статистическите качества на генерираната случайна последователност в реално време.



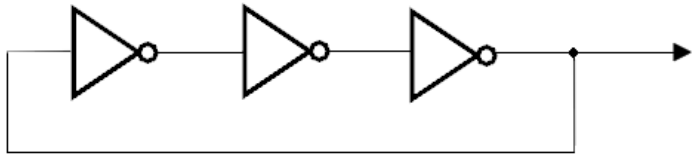

**Фиг. 1.5. Принципна схема на физически източник на шум**

На фиг. 1.5 е показана принципната схема на физически ентропиен източник, използващ шума на електронен прибор - в случая обратно-свързан стабилитрон, работещ в областта на пробива [21]. При такова свързване, върху стабилитрона се отделя шум с широка честотна лента (*~100* MHz). Предпочитат се стабилитрони с по-високи пробивни напрежения, тъй като генерираният шум е с по-голяма мощност. Стабилитронът *1N759* е с пробивно напрежение *12* V.

Променливотоковата съставяща се отделя с разделителния кондензатор *C1* и се усилва от двустъпален усилвател от типа *MAX2650 LNA* с ниско ниво на собствения шум. Общият коефициент на усилване е *40* dB, което е достатъчно изходния сигнал да се подаде към блока за цифровизация.

Би било полезно ентропийният източник да формира директно цифров сигнал. Известни са поне два такива принципа – състезания на логически сигнали и кръгови осцилатори. В основата и на двата принципа е непредсказуемостта на точното време за превключване на логическите елементи.

*Кръговият осцилатор* (Ring Oscilator) се състои от нечетен брой инвертори (фиг. 1.6). Периодът на осцилиране е равен на су-



**Фиг. 1.6. Схема на кръгов осцилатор**

мата от времената на превключване на инверторите. Периодът на осцилиране на схемата е непредсказуем, т.е. случаен, като следствие на непредсказуемостта на точното време за превключване на инверторите. Освен това, поради умишленото отсъствие на кварцова стабилизация, този период се влияе от странични фактори – промените в захранващото напрежение, промените на влажността, промените на температурата[4].

*RRNG* по схемата на кръговите осцилатори са от особен интерес - представляват ентропиен източник, близък до източник на бял шум, като директно формират цифров сигнал. Архитектурата *XS1* притежава четири такива осцилатора, всеки от които тактува собствен брояч[5] [38, 45]. Съдържанието на тези броячи не се изчиства при изключване или системен рестарт. Така се избягва връщането в предварително известна начална стойност и съответно се гарантира висока стойност на ентропията още от самото начало.

В статистическата механика под *ентропия* на системата се разбира величината

$$S = k \ln W, \tag{1.2}$$

където $k$ е константата на Болцман, а $W$ - статистическото тегло, т.е. броя различни микросъстояния (например различните възможни разположения на молекулите на газа), с които може да се реализира дадено макросъстояние (на газа в съда).

Основите на теорията на информацията са заложени от Колмогоров и Шенон. Едно от основните понятия в нея е ентропията. За равновероятни събития това понятие е въведено от Хартли през 1928 г. Клод Шенон уточнява понятието ентропия $H$ на случайната величина $\xi$, като го определя с израза [7, 12]

$$H(\xi) = -\sum_{i=1}^{n} p_i \log p_i \tag{1.3}$$

---

4 Поради отсъствието на каквато и да е форма на стабилизация на генерираната честота, тези генератори са известни като *генератори на свободни колебания* (Free Generators).

5 Първите представители на фамилията *XS1*, например *XS1-G4*, не притежават подобни генератори.

Ако се използва двоичен логаритъм, единицата за измерване на ентропията е *bit*, при натурален логаритъм - *nat*, а при десетичен логаритъм – *dit*. От (1.3) се вижда, че ентропията *H* на случайната величина $\xi$ в информационен смисъл представлява сума на вероятностите на отделните стойности на случайната величина $\xi$. Ентропията се разглежда като мярка за неопределеността на случайната величина, както и като мярка за информацията, необходима за описание на случайната величина.

Основните свойства на ентропията са следните:

$$H(\xi) \geq 0 \tag{1.4.1}$$

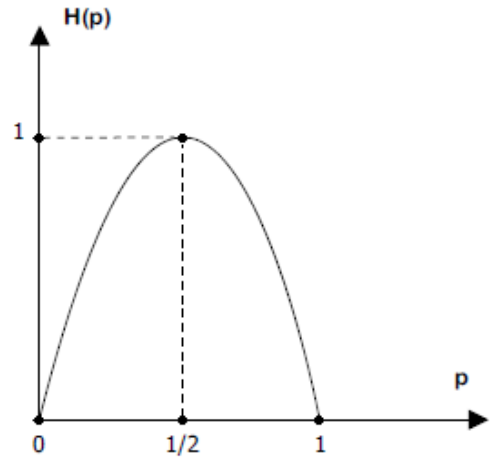
$$H(\xi) \leq \log n \tag{1.4.2}$$

$$H(\xi) = \log n \text{ при } p_i = \frac{1}{n}, i = 1,2, \dots n. \tag{1.4.3}$$

Изразите (1.4.1) и (1.4.2) определят съответно долната и горната граница на ентропията, докато (1.4.3) определя условието за достигане на максимума – всички *n* на брой стойности на случайната величина да са равновероятни.

При $n = 2$ нека обозначим с *p* вероятността $p_1$ на първата стойност, т.е. $p_1 = p$ и $p_2 = 1 - p$. След заместване в (1.3), се получава

$$H = -(p \log p + (1-p)\log(1-p)) \tag{1.5}$$




**Фиг. 1.7. Графика на ентропията при $n = 2$**

Графиката на функцията от израза (1.5) е показана на фиг.1.7. При $n = 2$ тя илюстрира основните свойства на ентропията: диапазо-

нът, максималната ѝ стойност от *1* bit, както и условието за достигане на максимума.

Максималната ентропия на случайна величина с $n = 2^{32}$ на брой стойности е *32* bit, което следва от (1.4.3). Ако обаче величината приема само *4* стойности, всяка с вероятност *0.25*, ентропията ѝ ще бъде само *2* bit, което се доказва с израза

$$H(\xi) = -\sum_{i=1}^{n} p_i \log p_i = -\sum_{i=1}^{4} \frac{1}{4} \log_2 2^{-2} = -4.\frac{1}{4}(-2) = 2 \text{ bit} \quad (1.6)$$

По отношение на физическите източници на шум се поставя изискването да формират сигнал с максимално възможна ентропия, т.е. сигналът им да бъде близък до белия шум, макар и в ограничен честотен диапазон.