VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Generátory pseudonáhodných čísel

Ročníkový projekt

27. dubna 2006 *Martin Fiala*

Prohlášení

Prohlašuji, že tento ročníkový projekt jsem vypracoval samostatně. Vycházel jsem z pramenů uvedených v seznamu použitých zdrojů a z konzultací s vedoucím ročníkového projektu**.**

……

Martin Fiala

27.dubna 2006

Poděkování

Děkuji vedoucímu ročníkového projektu panu Ing. Vladimíru Drábkovi za cenné rady a podněty poskytnuté při vypracování ročníkového projektu.

Martin Fiala

Abstrakt

Tento ročníkový projekt se zabývá návrhem, implementací a testováním dvou numerických generátorů pseudonáhodných čísel s modulem 2^k , které generují číselné hodnoty s rovnoměrným rozložením pravděpodobnosti. První je obvodovou realizací obecného lineárního kongruentního generátoru. Druhý pracuje na základě Fibonacciho (lagged) generátoru, jsou na něm ale ještě provedeny jisté úpravy. Oba generátory jsou implementovány v jazyce VHDL a jedná se tedy o jejich simulační modely.

Klíčová slova

generátor pseudonáhodných čísel, pseudonáhodná čísla, testování, lineární kongruentní metoda, Fibonacci lagged generator, modul 2^k, LCG, PRNG.

Obsah

1 **Úvod** **6**
1.1 Druhy generátorů……... 6
1.2 Cíl tohoto projektu…… 8
1.3 Teoretická východiska……... 9
1.3.1 Zbytková aritmetika……..... 9
1.3.2 Lineární kongruentní generátory (LCG)……... 9
1.3.3 Fibonacciho posloupnosti…….. .11
1.3.4 Využití vlastností modula 2^k při numerických výpočtech…. .12

2 Návrh generátorů **14**
2.1 Návrh lineárního kongruentního generátoru…….. 14
2.2 Návrh Fibonacciho (lagged) generátoru……... 20

3 Implementace **24**
3.1 Implementace lineárního kongruentního generátoru……. 24
3.2 Implementace Fibonacciho (lagged) generátoru……. 25
3.3 Ověřování výstupu generátorů……... 28

4 Závěr **29**

5 Použitá literatura a zdroje informací **30**

6 Přílohy **31**
6.1 Testování generátorů…… 31
6.1.1 Úvod……... 31
6.1.2 Empirické testy……... 32
6.1.3 Teoretické testy……... 38

1.ÚVOD

Generátory pseudonáhodných čísel (dále PRNG z anglického pseudorandom number generators) nacházejí využití na mnoha místech. Obecně všude tam, kde je nutné nebo vhodné simulovat nějaké náhodné jevy nebo procesy. Konkrétněji se jedná např. o oblasti šifrování a kryptografie, diagnostiky, telekomunikačních technologií, statistických výzkumů nebo počítačových simulací. Mohou být potřebné ve velkém množství počítačových aplikací, např. v oblasti grafiky, zpracování signálů, umělé inteligence, jako součást algoritmů některých numerických metod, ad.
Je zapotřebí poukázat na rozdíl mezi pojmy generátory náhodných čísel a generátory pseudonáhodných čísel. První jmenované vytvoří za nekonečně dlouhou dobu všechny posloupnosti všech délek, zatímco druhé budou generovat stále stejnou posloupnost pevné konečné délky. Na rozdíl od řady skutečně náhodných čísel je každé číslo pseudonáhodné posloupnosti determinované předchozím číslem a PRNG pracují na základě určitých matematických vztahů.
Generátory obyčejně produkují hodnoty v rovnoměrném rozložení pravděpodobnosti a v určitém intervalu, jehož „šířka“ závisí na modulu generátoru. Rovnoměrné rozložení v intervalu R<0,1) se nazývá normalizované rovnoměrné rozložení. Z rovnoměrného rozložení je poté možné pomocí odpovídajících transformací vytvořit požadované jiné rozložení (normální, exponenciální, ad.).
Důležitým pojmem souvisejícím s PRNG je modul generátoru. Modul určuje množinu čísel, z níž se budou vybírat čísla do generované posloupnosti.
Požadavky kladené na generátory závisejí na jejich konkrétním užití, ale obecně spočívají zejména v rychlosti generování, v délce periody generované posloupnosti a v co nejlepším přiblížení se vlastnostem skutečně náhodných posloupností. Univerzální PRNG vyhovující nárokům každého zájemce nebyl zatím nalezen, neboť vzhledem k deterministickému charakteru PRNG lze ke každému z nich najít aplikaci, pro niž generovaná posloupnost nebude dostatečně dobrá.

1.1 Druhy generátorů

Tato kapitola obsahuje stručný popis principů, na kterých jsou založeny v současnosti užívané generátory náhodných čísel.

Fyzikální generátory

Jsou obvykle realizovány pomocí nějakého zdroje skutečné náhodnosti v podobě fyzikálních jevů. Tyto generátory jsou schopny generovat opravdu náhodné (ne pseudonáhodné) posloupnosti.

Typický způsob realizace je pomocí zdroje šumu (šumového napětí), vzorkovacího obvodu, který měří okamžitou hodnotu napětí a analogově digitálního převodníku. V poslední době využívají často fyzikálního jevu radioaktivního rozpadu.
Ve srovnání s ostatními typy generátorů jsou obtížněji realizovatelné, ale také generují kvalitnější výstup.

Posuvné registry s lineární zpětnou vazbou (LFSR)

Matematické základy tohoto způsobu generování náležejí do teorie cyklických kódů. Obvody realizují dělení určitým binárním, tzv. generujícím polynomem. V každém kroku se odečte od aktuální hodnoty v registrech generující polynom.
Pro dosažení nejdelší periody se užívají jako generující polynomy nerozložitelné dvojkové polynomy, zvané primitivní.
N stupňový LFSR generátor může generovat posloupnost délky až 2^N-1.
Tyto generátory se často užívají v šifrování a kryptografii, zejména pro svou rychlost a délku periody a protože generují bitovou posloupnost (mají modul 2). Dále nacházejí uplatnění v diagnostice při vytváření testovacích kombinací pro obvod, kde není kvalita posloupnosti z hlediska náhodnosti hlavním požadavkem. Tím je vygenerování všech možných k-bitových binárních čísel pro daný počet bitů k.

Numerické generátory

Tyto generátory jsou obecně definovány matematickými rekurentními vztahy.

<u>Lineární kongruentní generátory (LCG).</u>

Vztah mezi dvěmi po sobě jdoucími pseudonáhodnými čísly je definován vztahem:

$$x_{i+1} = (x_i a + c) \bmod m,$$

kde x_i je pseudonáhodné číslo v iteraci i a x_{i+1} je pseudonáhodné číslo v následující iteraci, *a* se nazývá multiplikativní konstanta, c se nazývá aditivní konstanta a m je modul. Konstanty jsou zvoleny tak, aby dosahovala perioda generátoru co největší délky. Maximální možná délka posloupnosti je rovna m. Užívají se zejména jako programové generátory.

<u>Fibonacciho zpožděné (lagged) generátory.</u>

Tyto generátory nejsou tak rozšířeny jako LCG generátory, protože pro jejich použití je třeba mít již určitý počet pseudonáhodných čísel k dispozici a i v průběhu výpočtu je nutné uchovávat stejný počet vypočtených hodnot.

Generovaná posloupnost je definována vztahem :

$$x_{i+1} = (x_{i-c1} + x_{i-c2}) \bmod m, \quad i \geq \max(c1, c2) .$$

Na hodnotách konstant c_1 a c_2 závisí délka periody. Jsou známy hodnoty které zajišťují dlouhé periody.

Tento projekt je zaměřen především na tyto dvě skupiny generátorů a dále se jimi budeme zabývat podrobněji.

Ostatní typy numerických generátorů.

Ke kongruentním a Fibonacciho generátorům existuje řada příbuzných numerických generátorů. Jedná se o generátory původně anglicky zvané multiply-with-carry, add-with-carry, subtract-with-borrow, MGR, ad. Slibná budoucnost je předpovídána zejména generátorům typu multiply-with-carry.

Pozn.
V minulosti se pro získání náhodných čísel používaly mechanické generátory (např. kostky a rychle se otáčející kotouče), fyzikální generátory (na principu zářič - detektor) nebo tabulky náhodných čísel. Některé z nich nacházejí stále užití i v současnosti.

1.2 Cíl tohoto projektu

Cílem tohoto ročníkového projektu byl návrh a implementace generátoru pseudonáhodných čísel s modulem 2^k nebo 2^k-1 a následné experimentování s modelem. Cíl je tedy velmi obecný. Dle pokynů vedoucího ročníkového projektu jsem se zaměřil na obvodovou realizaci LCG. Při studiu odborné literatury jsem poté nalezl ještě i jiné numerické generátory vhodné pro hardwarovou implementaci.

1.2.1 Čeho bylo nakonec dosaženo.

Navržen byl nejprve universální obvodový model LCG generátoru, tzn. obvod s libovolně nastavitelnými parametry a, c a x_0. Ten byl poté implementován pro modul m = 2^{20} v jazyce VHDL. Implementovaný model je snadno upravitelný i pro jiné hodnoty modulu.
V další fázi byl navržen druhý numerický generátor pracující na základě Fibonacciho zpožděného (lagged) generátoru. Mým hlavním cílem při jeho návrhu byla rychlost generování. Nakonec se podařilo jej navrhnout i tak, aby úspěšně prošel všemi mně dostupnými testy. Podrobnosti lze nalézt v kapitole věnované návrhu. Implementována byla verze generátoru s parametry c_1 = 37, c_2 = 100, m = 2^{32}.
Předpokládaná funkčnost modelů byla potvrzena pomocí simulací v programu ModelSim.

1.3 Teoretická východiska.

Na efektivitu návrhu generátoru má zásadní vliv znalost některých teoretických poznatků týkajících se kongruentních generátorů, zbytkové aritmetiky, náhodných posloupností. V této kapitole jsou shrnuty ty, které byly při návrhu využity.

1.3.1 Zbytková aritmetika

Při úpravách vztahů pro kongruentní posloupnosti se často užívají některé axiomy modulární aritmetiky. Na tomto místě uvedu pouze způsob zápisu jednoho tvrzení, který nepovažuji za všeobecně známý.

Tvrzení : O dvou přirozených číslech n a m může být řečeno že jsou si rovna nebo rovna modulo d, když jejich zbytky po celočíselném dělení čílem d jsou si rovny.
Toto tvrzení můžeme zapsat $n = m,$ modulo d .

1.3.2 Lineární kongruentní generátory (LCG)

Vztah mezi dvěmi po sobě jdoucími pseudonáhodnými čísly je definován vztahem:

(1) $x_{i+1} = (x_i a + c) \bmod m$,

kde x_i je pseudonáhodné číslo v kroku i a x_{i+1} je pseudonáhodné číslo v následujícím kroku, a se nazývá multiplikativní konstanta, c se nyzývý aditivní konstanta a m je modul.
Tento vztah byl uveden již v úvodním přehledu generátorů, ale pro lepší přehlednost textu jsem jej zopakoval.

<u>Teorém A.</u>

Lineární kongruentní sekvence definovaná pomocí m, a , c a x_0 má periodu délky m když a jen když jsou splněny všechny následující podmínky

i) c je nesoudělné s m;
ii) $b = a - 1$ je násobek p, pro každé prvočíslo p, kterým je možné celočíselně vydělit m;
iii) b je násobek 4, jestliže m je násobek 4.

Důkaz teorému A je možné najít např. v [1].

Pro případ $c = 0$ platí ještě další teorémy, které jsou klíčové pro zjištění vhodného a se zajištěním generování periody délky m, ale pro jejich vysvětlení by bylo nutné uvést ještě velké množství definic a teoretických pojmů a protože existuje dostatečné množství vyhovujících multiplikátorů a i pro $c \neq 0$, nebudeme je zde uvádět.

Budeme-li mít multiplikátor tvaru $a = z^k + 1$ a $m = z^e$, kde z je základ číselné soustavy, e je přirozené číslo a $2 \le k < e$, potom a bude splňovat teorém A. Dle teorému A můžeme použít jako hodnotu c např. $c = 1$.

Tvar rekurentního vzorce (1) potom bude

(2) $$x_{i+1} = ((z^k + 1)x_i + 1) \bmod z^e.$$

Generátor realizovaný podle tohoto vztahu by nevyžadoval operaci násobení, ale pouze bitový posuv a sečtení a umožňuje tedy efektivní implementaci.
V logických systémech užíváme binární soustavu, čili vztah (2) pro tento případ nabude podoby

(3) $$x_{i+1} = ((2^k + 1)x_i + 1) \bmod 2^e.$$

Modul tohoto generátoru je roven 2e, což je ve shodě se zadáním ročníkového projektu, čili se nabízí jako vhodný k implementaci.
Generátory tohoto typu byly široce diskutovány v odborné literatuře a doporučovány mnoha autory. Přesto ukázaly již počáteční roky experimentování s touto metodou, že generátorů, vytvořených na základě (3) by se nemělo užívat, protože generovaná čísla nejsou dostatečně náhodná.

<u>Potence lineární kongruentní posloupnosti.</u>

V anglické odborné literatuře [1] se tento pojem píše potency, v tomto textu mu budu říkat potence.
Potence lineární kongruentní posloupnosti s maximální periodou je definována jako nejmenší celé číslo s takové, že

$$b^s = 0, \quad \text{modulo m.}$$

Takové celé číslo bude vždy existovat, když násobitel splňuje podmínku teorému A, že b je násobek každého prvočísla dělícího m.

D. Knuth označuje v [1] za podmínku dosažení dostatečně náhodných hodnot potenci o velikosti nejméně 5. V [1] je uveden příklad, ve kterém je ukázáno, že pro generátory (2) a modul $m = 2^{35}$ je uspokojivé potence dosaženo pro hodnoty $k \le 8$. To ale vede na „příliš malé“ koeficienty a, které se také v praxi neosvědčili. To tedy znamená, že pro hodnoty modulu $m \le 2^{35}$ nelze vytvořit dobrý generátor typu (2).

Další důležitý fakt, který by mohl být potenciálně využitelný při návrhu je tvrzení, že pro modul $m = 2^e \ge 8$ je dosaženo maximální potence, jestliže $a \bmod 8 = 5$. Důkaz tohoto tvrzení je možné nalézt v [1].
Potence je nutná, ale ne postačující podmínka pro dobrou náhodnost. Užívá se hlavně pro eliminaci parametrů, s kterými není možné vytvořit dostatečně náhodný generátor. Přijatelný LCG musí úspěšně projít spektrálním testem.
Podrobnější výklad o potenci je opět např. v [1].

1.3.3 Fibonacciho posloupnosti

Klasická Fibonacciho posloupnost je definována vztahem :

$$x_{i+1} = (x_i + x_{x=1}) \bmod m,$$

ale tato posloupnost není dostatečně náhodná, proto se užívá Fibonacciho zpožděná posloupnost (v původním znění Fibonacci lagged sequence) definovaná vztahem:

$$(4)\ x_{i+1} = (x_{i-c1} + x_{i-c2}) \bmod m,\ \ i \geq \max(c_1, c_2)\,.$$

Na hodnotách konstant c_1 a c_2 závisí délka periody. Na začátku výpočtu je třeba mít $\max(c_1, c_2)$ počátečních hodnot. V každé ireaci i je nutné mít k dispozici hodnoty x_{i-c1} a x_{i-c2}, proto musíme mít i v průběhu výpočtu $\max(c_1, c_2)$ paměťových míst pro uložení číselných hodnot x_{i-1} až $x_{i-\max(c1,c2)}$.

Pozn. Zápisem $\max(c_1,c_2)$ je míněna větší z dvojice hodnot c_1 a c_2.

Některé hodnoty konstant zapsané (l, k), kde k = $\max(c_1,c_2)$ a l je ta zbývající, které umožňují dosáhnout periody o délce $(2^k-1)2^{e-1}$ pro modul m = 2^e, jsou uvedeny v následující tabulce:

(24, 55)	(37, 100)	(83, 258)	(273, 607)	(576, 3217)	(7083, 19937)
(38, 89)	(30, 127)	(107, 378)	(1029, 2281)	(4187, 9689)	(9739, 23209)

tab. 1

Konstanty l,k jsou určeny tak, aby bity posloupnosti s nejnižší důležitostí (mod 2) měli periodu ($2^k - 1$). Potom i posloupnost musí mít periodu minimálně tak dlouhou. Důkaz vlastnosti, že při m = 2^e dostaneme posloupnost s délkou periody $(2^k-1)2^{e-1}$ zde uvádět nebudeme z důvodu jeho rozsáhlosti. Podrobně popsaný jej lze nalézt v [1 ,ex.30, str. 40]. Činnost generátoru je posána následujícím algoritmem.

<u>Algoritmus A pro (24, 55) .</u>

Paměťové buňky Y[1], Y[2], ... , Y[55] jsou na začátku nastaveny na hodnoty $x_{54}, x_{53}, \ldots, x_0$, j je nastaveno na 24 a k na 55. Postupným vykonáváním tohoto algoritmu budou produkovány čísla $x_{55}, x_{56}, \ldots$ jako výstup.

A1.

Y[k]:= (Y[k] + Y[j]) mod 2^e
Y[k] → výstup

A2.

```
j := j-1
k := k-1
if (j = 0) then j := 55 ;
else if (k=0) then k := 55;
```

Generátor podává výborné výkony u mnohých testů a jeho perioda je relativně dlouhá (v porovnání s LCG). Špatné výsledky vykazuje u testu, který přistupuje k výstupu generátoru jako k binární posloupnosti a provádí náhodnou cestu ve dvojrozměrném prostoru, na základě příchodu čísla 0 nebo 1. Aby generátor tímto testem prošel, je třeba provést v konečné fázi výpočtu následující úpravu.

<u>Úprava Martina Lüschera.</u>

Vygeneruje se určitý počet prvků Fibonacciho zpožděné posloupnosti (např. 500), tomuto množství budu dále říkat dávka a značit jej b (batch). Na výstup se ale umístí pouze několik prvků z počátku dávky (např. 50), označme jej p. Až se tyto prvky zkonzumují, vygeneruje se nových 500 prvků, opět se zkonzumuje pouze prvních 50 atd. Tímto opatřením se sníží korelace mezi hodnotami posloupnosti, které obvod finálně generuje. Tato metoda obecně zlepšuje náhodnost posloupností i při použití i u ostatních numerických generátorů. Je inspirována teorií chaosu. Jejím autorem je Martin Lüscher (Computer Physics Communications [1994]).

Dalším ze známých testů, s kterými má generátor bez provedení právě uvedené úpravy potíže, je test nazvaný (v původním znění) Birthday spacings test (6.1.2). Tento test představil v roce 1984 G. Marsaglia.

1.3.4 Využití vlastností modula 2^k při numerických výpočtech.

Každá z hodnot x_i vypočítaných podle vzorce (1) bude zřejmě mít v binární soustavě bitový rozsah od 2^0 do 2^{k-1}, tedy počet potřebných bitů pro vyčíslení hodnoty bude k. Operace modulo 2^k provedená nad jakýmkoli binárním číslem má tu vlastnost, že její výsledek je roven binární hodnotě, která je tvořena bity toho čísla v rozsahu odpovídajícímu řádům od 2^0 do 2^{k-1} . Hodnotu x_{i+1} z (1) tedy tvoří ze součinu $x_i a$ ta část této hodnoty, která je menší než 2^k, což je hodnota tvořená bity, v již zmíněném rozsahu od 2^0 do 2^{k-1}.

Situaci se pokusíme zobrazit následující ilustrací:

Hodnota součinu $x_i a$ ve vzorci (1), vypadá v binární soustavě jako posloupnost :

2^{k+n}	2^{k+n-1}		2^{k-1}	2^{k-2}	2^{k-3}	2^{k-4}		2^3	2^2	2^1	2^0
B_{k+n}		…..	B_{k-1}	B_{k-2}	B_{k-3}	B_{k-4}	..…	B_3	B_2	B_1	B_0

, kde Bj značí jété bity pro j od k+n do 0. Pro n platí, že n > 0 a n je přirozené číslo. Konkrétní hodnota n není pro náš závěr důležitá.
Každé z čísel vytvořené pomocí bitů reprezentujících řády v rozsahu od 2^k do 2^{k+n} je dělitelné číslem 2^k.

Jednoduchý důkaz:

$2^{k+n} + 2^{k+n-1} + \ldots + 2^{k+1} + 2^k \Rightarrow 2^k(2^n + 2^{n-1} + 2^{n-2} + \ldots + 2 + 1).$

Hledaná hodnota zbytku tedy připadá v úvahu pouze v bitovém rozsahu odpovídajícímu řádům od 2^0 do 2^{k-1}. Z vlastností pozičního způsobu zápisu čísel dále platí

$$(\sum_{0}^{k=1} 2^k) < 2^k .$$

Zbytek je tedy tvořen právě všemi bity odpovídajícími řádům v rozsahu od 2^0 do 2^{k-1}. Pro výpočet x_{i+1} tedy **stačí provést vynásobení** $x_i a$ **pouze v rozsahu bitů odpovídajících řádům od 2^0 do 2^{k-1}**. Tím dostaneme stejný výsledek, jako při provedení operace modulo 2^k s celou hodnotou $x_i a$.
Této vlastnosti se využívá v praxi běžně a v literatuře se považuje za samozřejmost a není blíže vysvětlena, ale jelikož mě nebyla okamžitě zřejmá a bude jí využíváno při implementaci, raději jsem ji zde popsal.

2.Návrh generátorů.

2.1 Návrh lineárního kongruentního generátoru.

Tento generátor je vlastně obvodovou realizací lineárního kongruentního generátoru (vztahu (1)) s libovolnou možností nastavení všech parametrů. Jedná se tedy o universální hardwareový model LCG. Kombinační obvod generátoru, provádějící numerický výpočet byl navržen se snahou docílit, co nejnižšího zpoždění obvodu.

Nejprve se pokusím znázornit, jaké numerické operace přesně vlastně obvod provádí. Známé schéma násobení dvou binárních čísel A, B (pro jednoduchost obě čtyřbitová) je:

```
                          A3 A2 A1 A0
                          B3 B2 B1  B0
 -------------------------------------------
              ( A3   A2   A1   A0 ) B0
         ( A3  A2   A1   A0 ) B1
    ( A3 A2  A1   A0 ) B2
( A3 A2 A1  A0 ) B3
-------------------------------------------
 C7  C6   C5   C4   C3   C2   C1  C0
```

Pro výsledek je potřeba osm bitů.

Schema operace $AB \bmod 2^4$ je podle vlastností popsaných v sekci 1.3.4:

$$AB \bmod 2^4$$

```
 -------------------------------------------
      ( A3   A2   A1   A0 ) B0
      ( A2   A1   A0 ) B1
      ( A1   A0 ) B2
      ( A0 ) B3
-------------------------------------------
        C3      C2      C1      C0
```

Pro výsledek jsou potřeba čtyři bity.

Analogicky ke druhému schematu můžeme vytvořit schema vztahu $y = (xa + c) \bmod 2^{20}$, na jehož základě funguje implementovaný generátor.

$$(xa + c) \bmod 2^{20}$$

---

c_{19} c_{18} c_{17} c_{16} c_{15} c_{14} c_{13} c_{12} c_{11} c_{10} c_9 c_8 c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0

$x_{19}a_0$ $x_{18}a_0$ $x_{17}a_0$ $x_{16}a_0$ $x_{15}a_0$ $x_{14}a_0$ $x_{13}a_0$ $x_{12}a_0$ $x_{11}a_0$ $x_{10}a_0$ x_9a_0 x_8a_0 x_7a_0 x_6a_0 x_5a_0 x_4a_0 x_3a_0 x_2a_0 x_1a_0 x_0a_0

$x_{18}a_1$ $x_{17}a_1$ $x_{16}a_1$ $x_{15}a_1$ $x_{14}a_1$ $x_{13}a_1$ $x_{12}a_1$ $x_{11}a_1$ $x_{10}a_1$ x_9a_1 x_8a_1 x_7a_1 x_6a_1 x_5a_1 x_4a_1 x_3a_1 x_2a_1 x_1a_1 x_0a_1

$x_{17}a_2$ $x_{16}a_2$ $x_{15}a_2$ $x_{14}a_2$ $x_{13}a_2$ $x_{12}a_2$ $x_{11}a_2$ $x_{10}a_2$ x_9a_2 x_8a_2 x_7a_2 x_6a_2 x_5a_2 x_4a_2 x_3a_2 x_2a_2 x_1a_2 x_0a_2

$x_{16}a_3$ $x_{15}a_3$ $x_{14}a_3$ $x_{13}a_3$ $x_{12}a_3$ $x_{11}a_3$ $x_{10}a_3$ x_9a_3 x_8a_3 x_7a_3 x_6a_3 x_5a_3 x_4a_3 x_3a_3 x_2a_3 x_1a_3 x_0a_3

$x_{15}a_4$ $x_{14}a_4$ $x_{13}a_4$ $x_{12}a_4$ $x_{11}a_4$ $x_{10}a_4$ x_9a_4 x_8a_4 x_7a_4 x_6a_4 x_5a_4 x_4a_4 x_3a_4 x_2a_4 x_1a_4 x_0a_4

$x_{14}a_5$ $x_{13}a_5$ $x_{12}a_5$ $x_{11}a_5$ $x_{10}a_5$ x_9a_5 x_8a_5 x_7a_5 x_6a_5 x_5a_5 x_4a_5 x_3a_5 x_2a_5 x_1a_5 x_0a_5

$x_{13}a_6$ $x_{12}a_6$ $x_{11}a_6$ $x_{10}a_6$ x_9a_6 x_8a_6 x_7a_6 x_6a_6 x_5a_6 x_4a_6 x_3a_6 x_2a_6 x_1a_6 x_0a_6

$x_{12}a_7$ $x_{11}a_7$ $x_{10}a_7$ x_9a_7 x_8a_7 x_7a_7 x_6a_7 x_5a_7 x_4a_7 x_3a_7 x_2a_7 x_1a_7 x_0a_7

$x_{11}a_8$ $x_{10}a_8$ x_9a_8 x_8a_8 x_7a_8 x_6a_8 x_5a_8 x_4a_8 x_3a_8 x_2a_8 x_1a_8 x_0a_8

$x_{10}a_9$ x_9a_9 x_8a_9 x_7a_9 x_6a_9 x_5a_9 x_4a_9 x_3a_9 x_2a_9 x_1a_9 x_0a_9

x_9a_{10} x_8a_{10} x_7a_{10} x_6a_{10} x_5a_{10} x_4a_{10} x_3a_{10} x_2a_{10} x_1a_{10} x_0a_{10}

x_8a_{11} x_7a_{11} x_6a_{11} x_5a_{11} x_4a_{11} x_3a_{11} x_2a_{11} x_1a_{11} x_0a_{11}

x_7a_{12} x_6a_{12} x_5a_{12} x_4a_{12} x_3a_{12} x_2a_{12} x_1a_{12} x_0a_{12}

x_6a_{13} x_5a_{13} x_4a_{13} x_3a_{13} x_2a_{13} x_1a_{13} x_0a_{13}

x_5a_{14} x_4a_{14} x_3a_{14} x_2a_{14} x_1a_{14} x_0a_{14}

x_4a_{15} x_3a_{15} x_2a_{15} x_1a_{15} x_0a_{15}

x_3a_{16} x_2a_{16} x_1a_{16} x_0a_{16}

x_2a_{17} x_1a_{17} x_0a_{17}

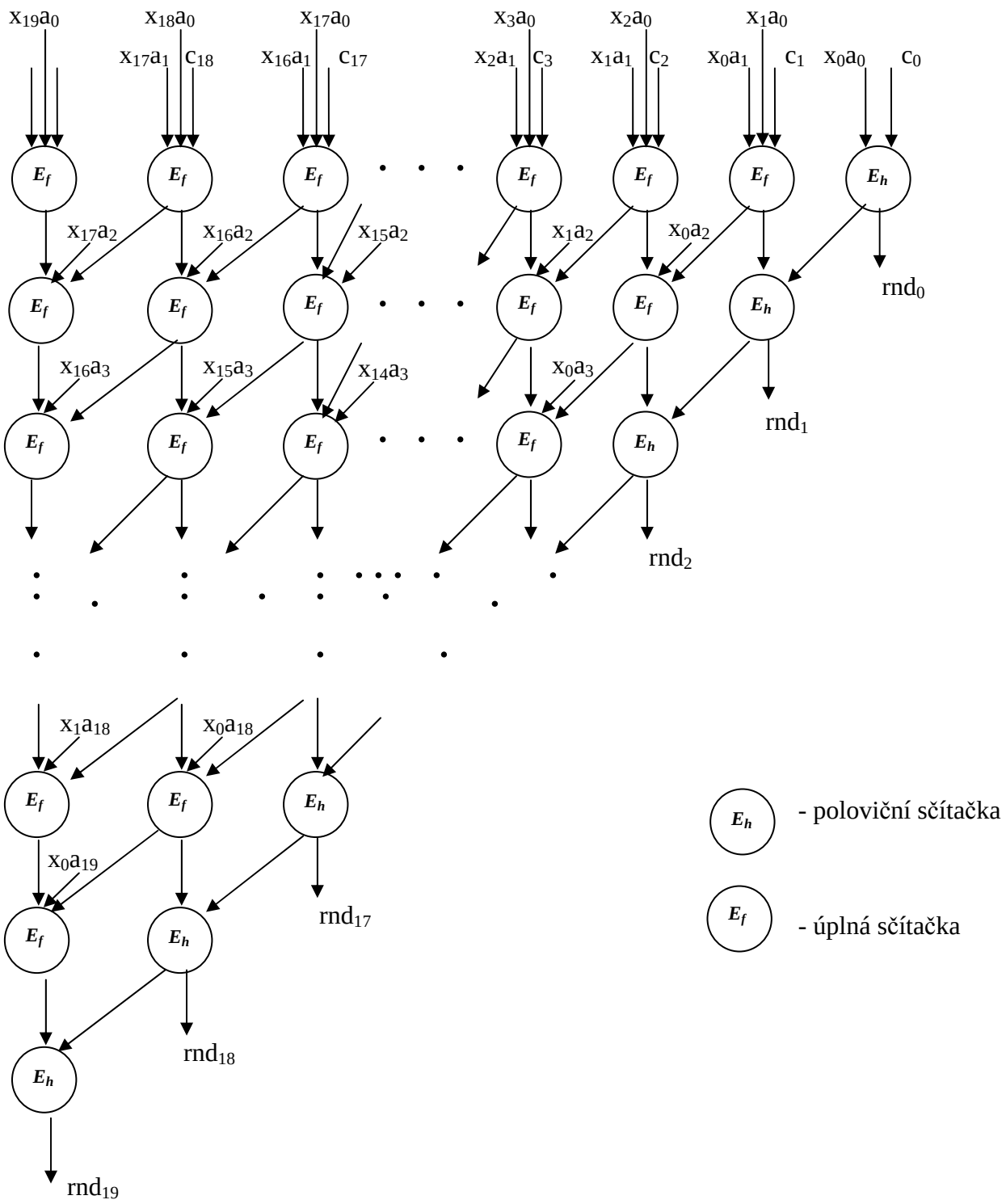
x_1a_{18} x_0a_{18}

x_0a_{19}

---

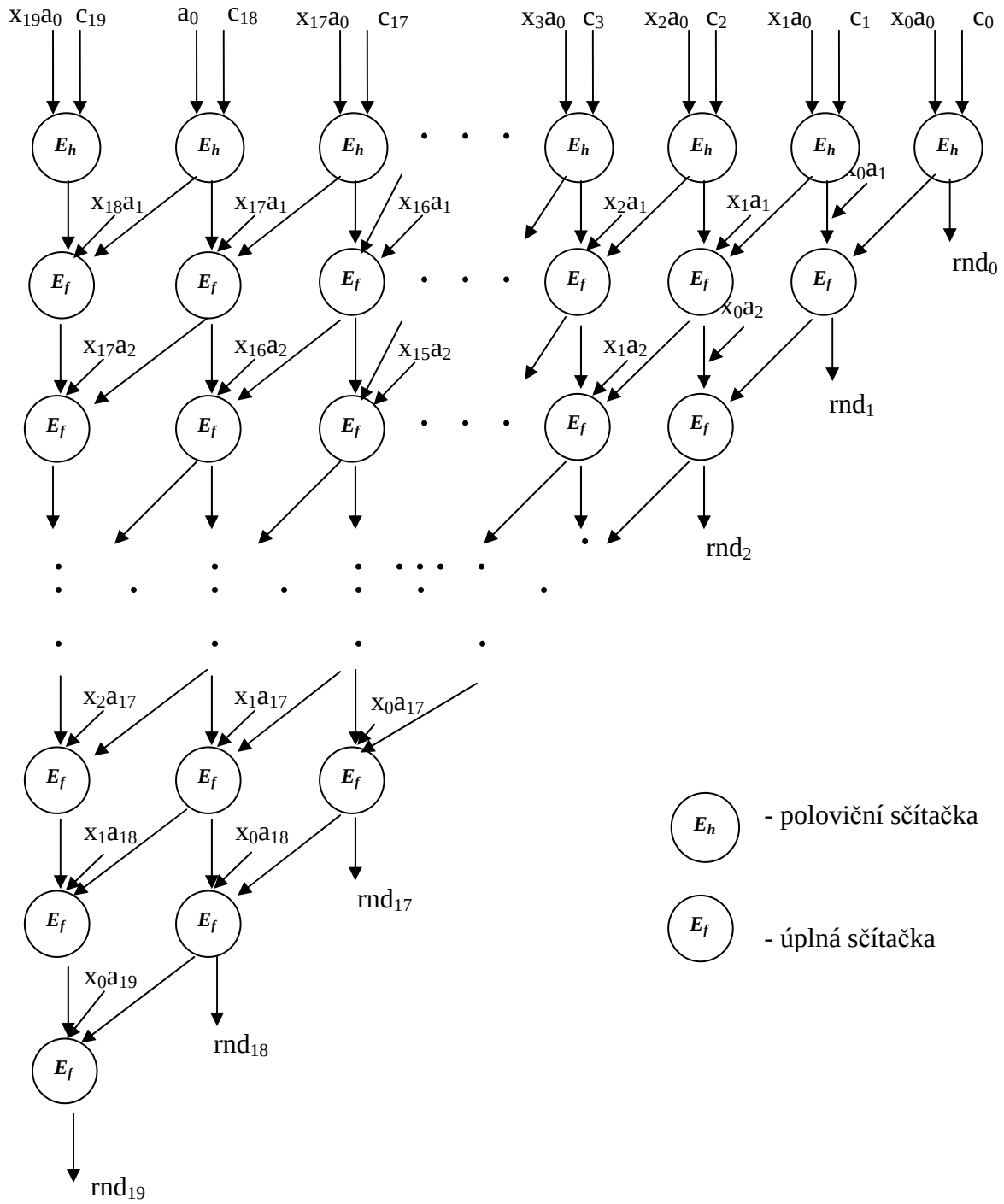
y_{19} y_{18} y_{17} y_{16} y_{15} y_{14} y_{13} y_{12} y_{11} y_{10} y_9 y_8 y_7 y_6 y_5 y_4 y_3 y_2 y_1 y_0

Obvod uskutečňující tento výpočet je realizován pomocí sčítaček zapojených způsobem Walaceova stromu, any bylo dosaženo co nejmenšího zpoždění.
Sčítačky mohou být rozvrženy dvěma způsoby, obojí zapojení je rovnocenné vzhledem ke zpoždění obvodu a k počtu logických členů. Zkrácená schemata obvodů jsou naznačena na následujících dvou obrázcích.



Obr.3 První možný způsob zapojení

Pozn.
Carry se předává vždy do vyššího řádu (doleva, šikmé linie), součty se předávají následující sčítačce ve stejném řádu (dolů, vodorovné linie).
V tomto zapojení jsou poloviční sčítačky v nejspodnější šikmé linii.



Pozn.
V tomto zapojení jsou poloviční sčítačky v první vodorovné řadě.

Označme T_c zpoždění signálu carry, T_{sha} zpoždění signálu součtu u jednobitové poloviční sčítačky, T_{sfa} zpoždění signálu součtu u jednobitové úplné sčítačky a T_{and} zpoždění hradla and. Zpoždění T obvodu je potom

$$T = (k-2)T_c + T_{sha} + T_{sfa} + T_{and}, \text{ v případě, že } T_c > T_s$$

nebo

$$T = kTs + T_{and}, \text{ pro } T_s > T_c.$$

Počet potřebných komponent L_{fa} (úplných sšítaček), L_{ha} (polovičních sčítaček) a L_{and} (dvouvstupových hradel and) je :

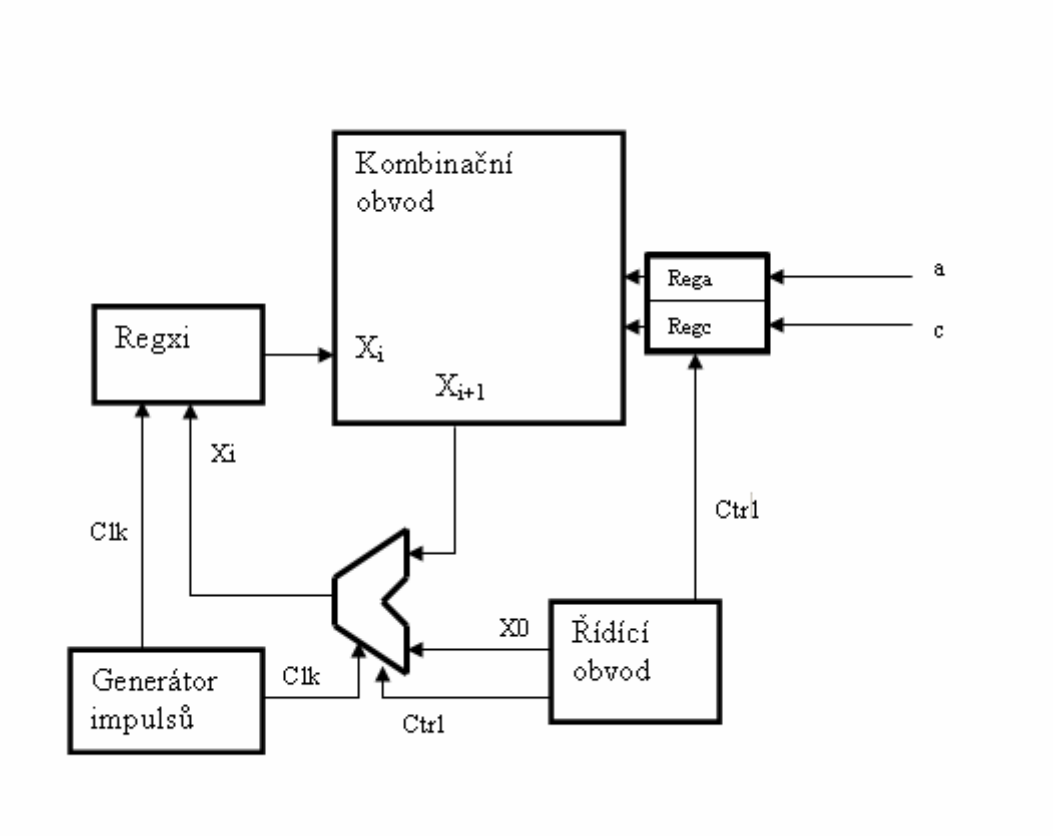
$L_{fa} = k(k-1)/2 = 190,$
$L_{ha} = k = 20,$
$L_{and} = k + k(k-1)/2 = 210.$

Kdybychom mezi jednotlivými úrovněmi 1 až k zavedli synchronizaci a umístili mezi ně registry, do kterých by se ukládali mezivýsledky, mohli bychom vytvořit řetězenou linku. Na začátku výpočtu by ale bylo nutné nějakým způsobem dodávat postupně na vstup obvodu prvních k (pro $m = 2^k$) členů generované posloupnosti náhodných čísel.

Budeme se ale dále zabývat neřetězenou verzí obvodu. Po dobu činnosti generátoru je nutné „podržet“ na vstupech kombinačního obvodu hodnoty konstant *a* a c. Dále je třeba mít mezi jednotlivými iteračními kroky uchovanou právě vypočtenou hodnotu x_{i+1}, která je v dalším kroku jednou ze vstupních hodnot. K uspokojení tšchto požadavků poslouží trojice registrů.

Na začátku výpočtu se nastaví jako vstupní hodnota x_i nějaká hodnota x_0, v dalších krocích se jako hodnota x_i použije výstupní hodnota kombinačního obvodu generátoru (x_{i+1}). K realizaci výběru mezi těmito hodnotami se použije multiplexor. Z výstupu kombinačního obvodu se musí odebírat hodnota až po uplynutí určité doby, po které se tam ustálí vypočtená správná hodnota. Pro měření této doby a pro synchronizaci registru pro uložení vypočtené hodnoty x_i potřebujeme generátor hodinových impulsů.
Nazávěr je třeba ještě obvod, který bude řídit funkci multiplexoru a na začátku výpočtu nastaví správné hodnoty do registrů pro konstanty a, c a x_0.

Na obr.4 je blokové schema celého navrhovaného generátoru.



Obr.4 Blokové schema navrhovaného obvodu.

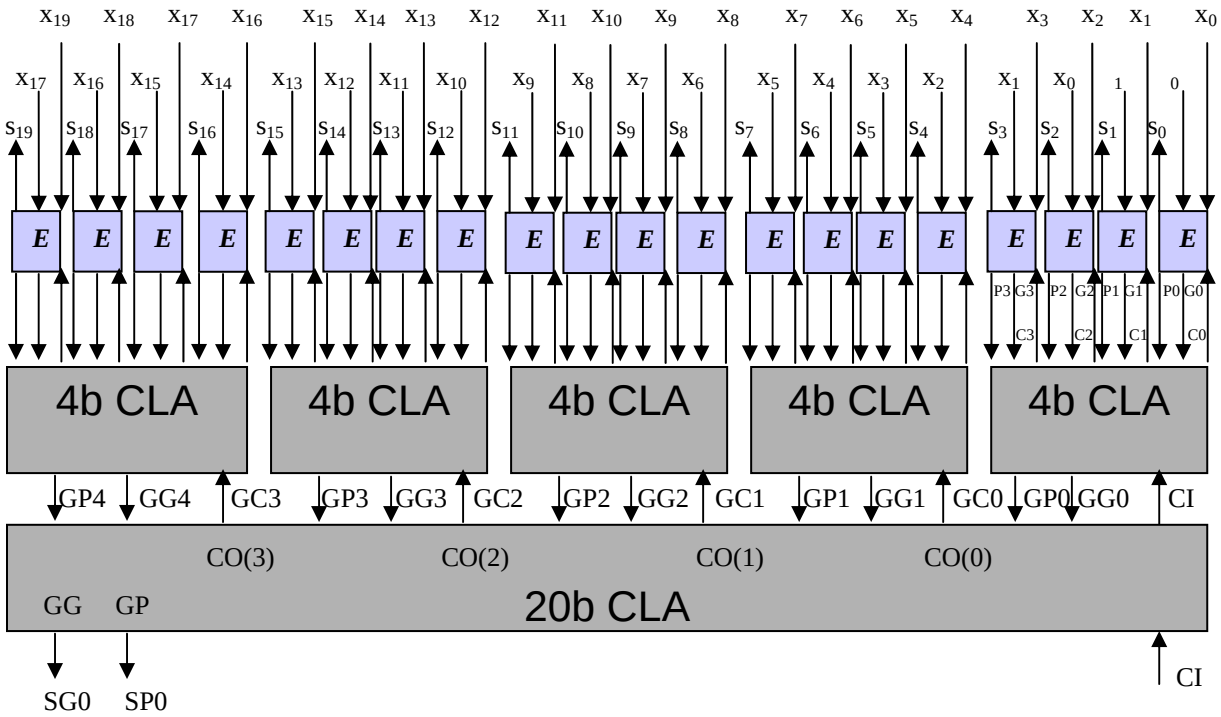
Pozn.
Komponentu řídícího obvodu by bylo v praxi vhodnější implementovat jiným způsobem, než na následujícím obr. tak, aby bylo možné měnit nebo aby se nějak měnili hodnoty a, c a x_i v průběhu výpočtu.

2.2 Návrh Fibonacciho (lagged) generátoru.

Tento generátor jsem se snažil navrhnout s cílem dosažení minimálního zpoždění obvodu, ale přesto i slušného výstupu generátoru. Původně se nemělo jednat o generátor Fibonacciho (lagged) posloupnosti, ale o jiný rychlejší LCG.
Způsob, jak vytvořit generátor s menším zpožděním jsem nejprve spatřoval v nalezení vhodných hodnot parametrů a a c pro LCG tak, aby byl výpočet realizován oproti předchozímu universálnějšímu řešení pomocí méně součtů. Nápomocné přitom byli zákonitosti mezi parametry popsané v sekci 1.3.2. Nakonec jsem ale žádné řešení, které by odpovídalo mým „představám“, nenalezl. Extrémní případ, který se nabízel je popsán vztahem (3, kap.1.3.2). Takovýto obvod by realizoval pouze součet vstupní hodnoty se vstupní hodnotou posunotou o několik míst doleva (posunutí závisí na konkrétní hodnotě parametru a). V podstatě by realizoval lineární kongruentní generátor pro nějaké pevně zvolené vyhovující konstanty a a c.

Pro dosažení nejmenšího zpoždění by byly použity sčítačky s CLA (carry look ahead), které vykazují podstatně menší zpoždění než ripple carry.

Na následujícím obr. je schema původně zamýšleného obvodu. Konstanty jsou zvoleny $a = 2^k + 1 = 5$, c = 1 a modul m=2^{20}.



Obr.5 Blokové schema původně zamýšleného generátoru.

Tento způsob generování je ale přípiš jednoduchý a v praxi se tyto generátory neosvědčili. Jejich výsledky ve spektrálních testech (6.1.3) jsou rovněž velmi špatné.

Sčítačku s CLA, která je základem právě prezentovaného obvodu je ale možné využít při návrhu Fibonacciho zpožděného (lagged) generátoru (1.3.3), který vykazuje mnohem lepší výsledky a je v praxi hojně užívaný. Připomeňme, že generátor pracuje na základě vztahu

$$x_{i+1} = (x_{i-c1} + x_{i-c2}) \bmod 2^k,$$

a tuto operaci je právě možné efektivně realizovat pomocí k-bitové sčítačky s CLA.

Abychom se přesvědčili, že bude tento obvod skutečně generovat kvalitní náhodné hodnoty a nenechali tento fakt na náhodě, je třeba jej otestovat nějakou sérií testů. K tomuto účelu se nabízí sada testů náhodnosti, nazvaná Diehard, kterou během svého dlouholetého působení na poli pseudonáhodných generátorů sestavil profesor George Marsaglia. Tuto sadu je možné volně stahovat z webové adresy http://www.stat.fsu.edu/pub/diehard/.
Jsou zde obsaženy testy nazvané (v původním znění) Birthday spacings test (6.1.2), The overlapping 5 – permutation test, Binary Rank test, The bitstream test, Overlapping-Pairs-Sparse-Occupancy, Overlapping-Quadruples-Sparse-Occupancy, DNA test, Count-the-1's test, Parking lot test, Minimum distance test, The 3d spheres test, Squeeze test, Overlapping sums test, Runs test (6.1.2) a Craps test. Podrobný popis testů je přiložen v souboru tests.txt.
Jak bylo předesláno, Fibonacciho generátory bez úpravy neprochází „Birthday spacings testem“. Ještě zde považuji za vhodné poznamenat, že jsem pomocí této baterie testů otestoval i několik „klasických“ LCG a většina podávala nedostatečné výkony (např. Ansic, Randu, Simscript, Bcpl, Urn 12, Crand, Apple a některé Fishmanovi a Hoaglinovi generátory).
Aby bylo možné generátor otestovat, bylo zapotřebí obstarat jeho dostatečně dlouhý výstup. Toho bylo docíleno implementací generátoru v C na základě návodu podle D. Knutha [1, str.186], kde je implementován Fibonacciho generátor na základě vztahu

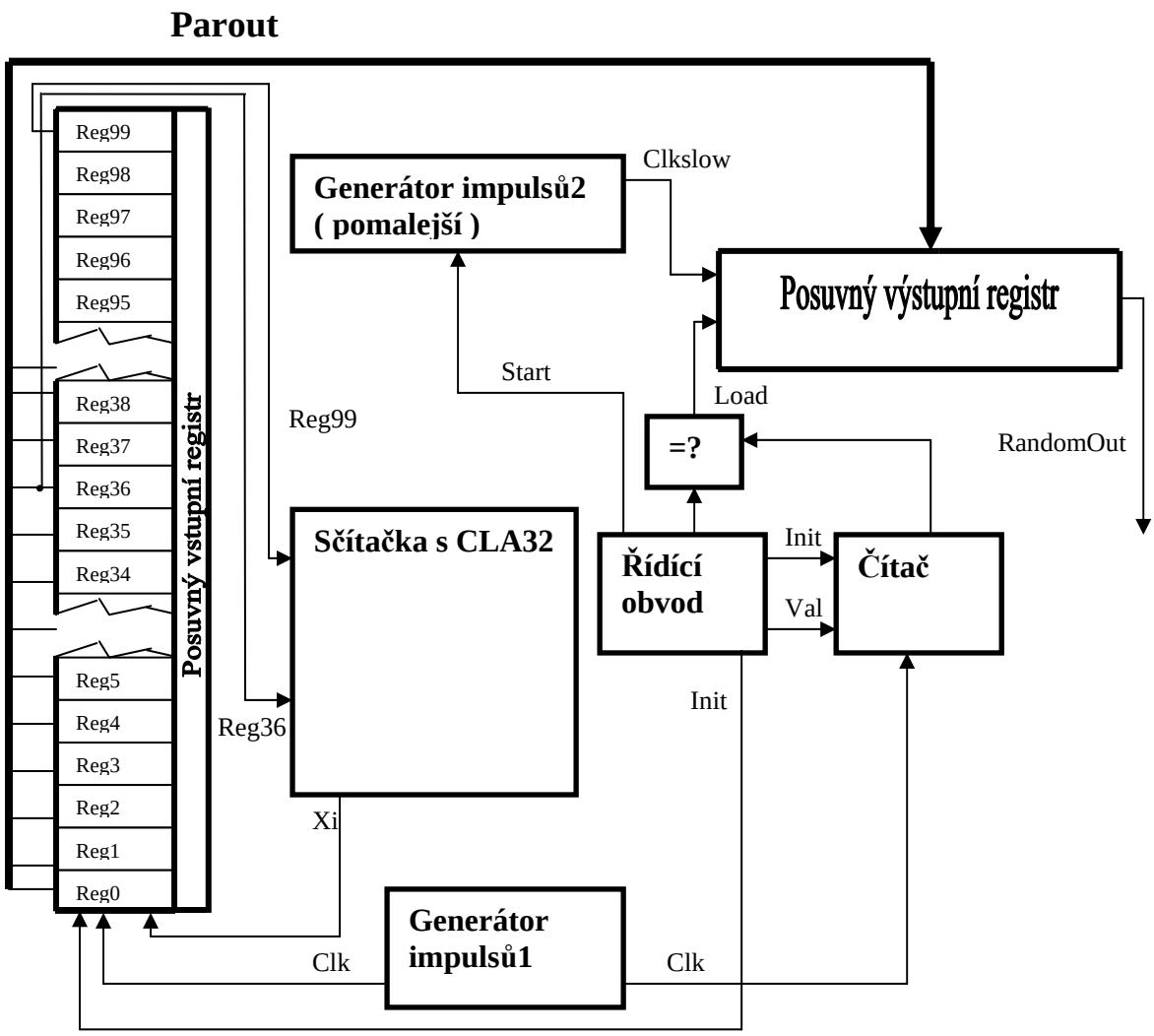
$$x_{i+1} = (x_{i-37} - x_{i-100}) \bmod 2^{30}.$$

Sada Diehard ale vyžaduje hodnoty v rozsahu 32 bitů a uvedený program neobsahuje úpravu podle Martina Lüschera (1.3.3) a realizuje odčítání dvou čísel místo sčítání. Dále jej bylo nutné upravit tak, aby vypisoval hodnoty podle formátu, který vyžaduje jedna z utilit sady Diehard, která převádí asci textový soubor na binární soubor, který lze již otestovat. Mnou vytvořená a použitá verze programového generátoru je přiložena.
Výsledky testů byly výborné, posloupnost úspěšně prošla všemi testy. Generátor lze tedy považovat za perspektivní pro realizaci.
Obvodová realizace Fibonacciho generátoru bez Lüscherovi úpravy je vlastně ekvivalentní obvodové simulaci algoritmu A uvedeného v kap. (1.3.3) pro odpovídající konstanty c_1 a c_2. Pro implementaci jsem zvolil variantu generátoru, pro kterou platí $c_1 = 37$, $c_2 = 100$ a pracuje na základě vztahu

$$x_{i+1} = (x_{i-37} + x_{i-100}) \bmod 2^{32}.$$

Pro usnadnění vysvětlení návrhu obvodu budu dále v textu uvažovat pouze tento konkrétní případ generátoru.
Kombinační obvod lze jak bylo výše naznačeno efektivně realizovat pomocí 32 bitové sčítačky s CLA.
V průběhu výpočtu je nutné uchovávat hodnoty spočtené v průběhu sta předchozích iterací a na vstup kombinačního obvodu přivádět hodnotu, která byla vygenerována „před sto iteracemi“ (i – 100) a hodnotu, která byla vygenerována „před sedmatřiceti (i – 37) iteracemi“. Tuto činnost bude provádět posuvný registr, jehož stý a sedmatřicátý paralelní výstup bude přiveden na vstup kombinačního obvodu. Po skončení výpočtu hodnoty x_i v aktuální iteraci se posune obsah posuvného registru o jeden stupeň směrem k poslednímu

registru (doprava) a právě vypočtená hodnota x_i se uloží do prvního stupně posuvného registru.
Posuvný registr je nutné v okamžiku zahájení činnosti obvodu zinicializovat na vhodných sto počátečních pomocných náhodných číselných hodnot. Podle Knutha [1, str.27] nesmí být všechny liché.
Označme b počet čísel, kterými bude tvořena jedna dávka čísel (dávka ve smyslu podle Lüscherovi úpravy) a p počet čísel z počátku dávky, které se umístí na výstup.
K realizaci generování hodnot takovým způsobem, aby se na výstup generátoru umístil jen počet hodnot $p < b$ a $p<100$, z dávky hodnot o délce b, vygenerovaných pomocí simulace vztahu (4), použijeme čítač a další posuvný registr. Pro rozlišení jej nazvěme výstupní posuvný registr a předchozímu říkejme vstupní posuvný registr.
Výstupní registr bude sestávat z p stupňů.
Čítač bude odpočítávat (inkrementovat se) jednotlivé iterace kombinačního obvodu. Po dosažení hodnoty, která odpovídá počtu hodnot dávky b se přepíše prvních p hodnot ze vstupního posuvného registru, do výstupního posuvného registru. Seriový výstup výstupního registru bude výstupem celého generátoru. Aby bylo možné generovat hodnoty rovnoměrnou rychlostí, je nutné, aby výstupní posuvný registr posunul svůj obsah p krát za stejnou dobu, za kterou vygeneruje kombinační obvod b hodnot (provede se b iterací).
Na následujícím obrázku je blokové schema právě popsaného generátoru.



Obr.5 Blokové schema pro modifikovaný Fibonacciho generátor.

Pozn.
Signál Parout zahrnuje hodnoty p „nejnižších“ registrů vstupního posuvného registru.

3. Implementace

Generátory jsou implementovány jako simulační modely v jazyce vhdl.

3.1 Implementace lineárního kongruentního generátoru.

Navržený generátor byl implementován pro modul m o velikosti $m = 2^{20} = 1048576$. Takový modul je pro použití v praxi příliš malý, ale umožňuje vygenerovat celou posloupnost a ověřit, zda je generována skutečně posloupnost o délce rovné předpokládané periodě. Při implementaci navíc vyplynulo, že kombinační obvod je možné implementovat pomocí cyklů, díky čemuž stačí pro získání modelu generátoru, který by generoval hodnoty na jiném počtu bitů, než je implicitních dvacet, změnit hodnotu příslušné konstanty v kódu.

Pohled na generátor jako celek

Systém jako celek zobrazený na obr.4 je relativně jednoduchý, resp. jeho funkčnost je podle mého názoru poměrně zřejmá a hardwarově realizovatelná bez nepředvídatelných obtíží, nepovažuji proto za nutné jej v modelu implementovat např. až na úroveň synchronizace mezi multiplexorem a registrem RegXi. Všechny registry jsou reprezentovány pouze signály v architektuře pro test bench, nikoli samostatnými komponentami. Přiřazením hodnoty signálu se tedy zapíše hodnota do registru. Komponentami testovací entity (test bench) je kombinační obvod generátoru, generátor hodinových impulsů a multiplexor. Řídící obvod je simulován nastavením signálů, které by jím byly nastavovány (hodnoty konstant a, c, x_0), v architektuře testovací entity na začátku výpočtu.

Kombinační obvod

Obvod je popsán strukturálně, bylo tedy nutné implementovat i komponenty, z kterých se skládá. Jedná se o poloviční jednobitovou sčítačku, úplnou jednobitovou sčítačku a logické hradlo and. Ty byly již popsány behaviorálně. Díky pravidelné struktuře obvodu jej bylo možné namodelovat pomocí cyklů. Komponenty jsou propojeny pomocí dvojrozměrného pole dvacetibitových signálů. Implementována byla první verze obvodu z obr.3.

Generátor hodinových impulsů

Běžný generátor hodinových impuplsů, který generuje hodinový signál clk. Délka jeho periody musí být minimálně rovna součtu doby výpočtu kombinačního obvodu (doby za kterou se na výstupu ustálí správná hodnota) a doby potřebné k tomu, aby se hodnota x_i dostala z výstupu kombinačního obvodu na jeho vstup.

Multiplexor

Je synchronizován pomocí hodinového signálu clk. V okamžiku náběžné hrany signálu clk se umístí na výstup multiplexoru buď hodnota signálu výstupu kombinačního obvodu, pokud je signál Ctrl nastaven na hodnotu 0 nebo hodnota x_0, pokud je signál Ctrl nastaven na hodnotu 1.

Doplňující poznámky.

Po spuštění simulace (v čase 0) se provedou inicializace hodnot signálů Ctrl, Rega, Regc a X0. Toho je docíleno jednorázovými přiřazeními (ne process) v těle architektury testovací entity. Hodnota jednobitového řídícího signálu Ctrl musí být v době, kdy se má přepsat první hodnota Xi z výstupu kombinačního obvodu do registru RegXi, nastavena na opačnou logickou hodnotu vzhledem k hodnotě, na kterou byl signál inicializován. Veškeré signály jsou typu bit a bit_vector.

3.2 Implementace Fibonacciho (lagged) generátoru.

Při implementaci Fibonacciho laged generátoru jsem se držel co nejvíce parametrů, které byly užity v programové implementaci generátoru, kterou jsem používal ve fázi návrhu. Smyslem mělo být snadnější ověření správnosti výstupu obvodového generátoru díky možnosti porovnání generovaných hodnot se softwarovým generátorem. Proto jsem použil parametry m, c_1 a c_2 se stejnými hodnotami a sice $m = 2^{32}$, $c_1 = 37$, $c_2 = 100$. Perioda, kterou by generoval čistě Fibonacciho generátor bez úprav by byla $(2^{100} - 1)2^{31}$. Délku dávky (batch) b jsem zvolil b = 500 a počet hodnot p (zachovám-li značení použité v kapitole 1.3.3), jsem zvolil p = 50.
Model není tak snadno upravitelný pro jiné hodnoty modulu jako model pro LCG. Bitový rozsah registrů v závislosti na délce slova, která v nich má být uložena lze změnit snadno, ale problém spočívá v nutnosti vyměnit stávající sčítačku s CLA za sčítačku s CLA s odpovídajícím bitovým rozsahem. Kdyby bylo možné implementovat ve vhdl sčítačku s CLA universálnějším způsobem nezávisle na její bitové šířce, např. jen v závislosti na několika konstantách, byl by obvod upravitelný snadno.

Pohled na celý systém

Činnost i struktura tohoto generátoru je již složitější, než tomu bylo u předcházejícího. Tento model obsahuje veškeré signály potřebné pro řízení a funkci všech komponent. Žádné pomocné signály nejsou zanedbány.
Generátor sestává z následujících komponent : dvojice generátorů hodinových impulsů, dvojice posuvných registrů (vstupní a výstupní), komparátor, čítač a kombinační obvod generátoru. Veškeré komponenty vyjma kombinační obvod generátoru jsou popsány pouze behaviorálně.

Implementace kombinačního obvodu

Kombinační obvod je popsán strukturálně, ačkoli se v tomto případě jedná o naprosto standardní 32-bitovou sčítačku s CLA. Komponenty, z kterých se skládá jsou FPGA, 4-bitová CLA. Poslední (třetí) úroveň CLA je využitá jen z poloviny a pro ušetření počtu součástek by ji mohlo být možno realizovat přímo pomocí hradel and a or, místo celé CLA.

Generátor hodinových impulsů1

Generuje hodinový signál clk. Délka jeho periody musí být minimálně rovna součtu doby výpočtu kombinačního obvodu a doby, za kterou se provede posun vstupního posuvného registru a hodnoty jeho 37. a 100. paralelního výstupu jsou propagovány na vstup kombinačního obvodu.

Vstupní posuvný registr

Posuvný registr obsahující sto 32-bitových registrů. Činnost komponenty je aktivována náběžnou hranou signálu clk. V závislosti na hodnotě signálu Init, kterým je posuvný registr řízen, se provede buď paralelní nastavení posuvného registru na hodnoty na paralelních vstupech, jestliže je hodnota signálu load rovna 1 nebo posuv registru a nastavení prvního registru posuvného registru na hodnotu na seriovém vstupu, je-li hodnota signálu load rovna 0. Paralelní výstup je k dispozici po celou dobu činnosti generátoru.

Čítač

Je synchronizován pomocí hodinového signálu clk. Při každé náběžné hraně signálu clk se inkrementuje o jedna, je-li aktuální hodnota menší než 500, nebo-li počet prvků dávky.
V jiném případě je nastaven na hodnotu 1. Řízen je pomocí signálu Init. Jestliže má tento signál hodnotu 1, nastaví se čítač na hodnotu vstupního signálu Val.

Komparátor

Slouží k signalizaci skutečnosti, že bylo v **předchozím taktu** dokončeno generování jedné dávky, či-li porovnává výstup čítače s hodnotou 500. V blokovém schematu je hodnotu, kterou komparátor porovnává, možné nastavit z řídícího obvodu, ale v implementovaném modelu je nastavena „na pevno“ v těle komparátoru. Tento rozdíl je způsoben tím, že jsem blokové schema kreslil až po implementaci a tento nedostatek modelu jsem již z časových dúvodů nemohl opravit.
Na začátku **tohoto taktu** byl (v okamžiku náběžné hrany signálu clk) zahájen posun vstupního posuvného registru a přepis hodnoty z výstupu kombinačního obvodu generátoru do vstupního posuvného registru. Je nutné zajistit, aby byl výstupní signál komparátoru nastaven na 1 až po aktualizaci (posunu) vstupního posuvného registru, protože komparátor je spouští pro přepis hodnot ze vstupního posuvného registru do výstupního posuvného

registru. Toho je dosaženo tím, že čítač a komparátor mají větší zpoždění než operace aktualizace vstup. reg.

Generátor hodinových impulsů2

Generuje hodinový signál clkslow. Tímto signálem je aktivován posun výstupního posuvného registru. Délka periody signálu musí být b/p krát (10 krát) větší než u signálu clk, aby bylo docíleno chování obvodu zamýšlené při návrhu. Generátor není aktivní od okamžiku nastartování činnosti celého obvodu, ale až od okamžiku, kdy je nastaven řídící signál Start na hodnotu 1.

Výstupní posuvný registr

Obsahuje padesát 32-bitových registrů. Návaznost k některým ostatním komponentám již byla popsána. Registr začne pracovat až v momentě započetí činnosti druhého hodinového generátoru. Paralelní nastavení obsahu posuvného registru je asynchronní a provede se v reakci na náběžnou hranu vstupního řídícího signálu load. Touto operací se přepíše prvních p hodnot ze vstupního posuvného registru do tohoto. Posuv registru je synchronní a provede se v reakci na náběžnou hranu signálu clkslow. Implementovaný obvod je nastaven tak, že je činnost výstupního registru nastartována poté, co komb. obvod vygeneruje prvních p hodnot . Takto je to provedeno z důvodu snahy o dodržení stejné funkce, jako u programového generátoru. Náhodné hodnoty je možné odebírat ze signálu RandomOut až po uplynutí určité doby od aktivace generátoru.

Doplňující poznámky

Řídící obvod je opět suplován jednorázovými příkazi v architektuře pro test bench. Signál Init je inicializován na hodnotu 1 a slouží k inicializaci čítače a vstupního posuvného registru. Po uplynutí poloviny periody signálu clk se nastaví na 0. Signál Start je inicializován na hodnotu 0 a po uplynutí doby T, pro kterou platí

$$T = (\text{doba periody clk})\text{b},$$

je jeho hodnota změněna na 1.
Všechny signály jsou typu bit a bit_vector, vyjma vstupních a výstupních datových signálů čítače a vstupních datových signálů komparátoru, které jsou kvůli usnadnění implementace čítače, typu std_logic_vector.

Pozn.
Názvy signálů a komponent ve vhdl kódu se vždy neshodují s názvy uvedenými v tomto dokumentu. Které identifikátory odpovídají názvům zde použitým je vysvětleno v přiložené programové dokumentaci.

Zpoždění základních logických členů a komponent je u obou modelů nastaveno pouze tak, aby odpovídalo ve vzájemném porovnání prvků, tzn. např. prvky s jednoúrovňovým zpožděním jsou nastaveny na zpoždění 2 ns, prvky s dvouúrovňovým zpožděním na 4 ns, atd.

3.3 Ověřování výstupu generátorů

Test benche obou generátorů obsahují process, který vypisuje v každém taktu aktuální náhodnou hodnotu do textového souboru. Posloupnosti v textových souborech je pak možné otestovat nebo ověřit, zda splňují předpokládané vlastnosti.

Lineární kongruentní generátor.

Pro ověření správné funkce jsem implementoval jednoduchý softwarový LCG v jazyce C a vždy porovnal shodnost několika počátečních a koncových hodnot posloupností vygenerovaných oběma generátory. Dále jsem ověřil i délku posloupnosti. Tento generátor by bylo vhodné ještě prověřit i testy náhodnosti a výsledky porovnat se sotwarovým generátorem se stejnými parametry, ale z časových důvodů jsem to již nebyl schopen uskutečnit.

Generátor Fibonacciho posloupností.

Záměr porovnat výsledky, které generuje implementovaný model se softwarovým generátorem se nepodařilo realizovat, protože jsem si nepovšiml jisté implementační odchylky.
Proto bylo nutné předpokládané výsledné hodnoty ověřit manuálním výpočtem, což bylo provedeno pro několik počátečních hodnot posloupnosti.
Výstupní hodnoty tohoto generátoru je již ale možné otestovat sadou Diehard. Pro otestování je vyžadován soubor s přibližně třemi miliony číselných hodnot. Doba simulace v programu ModelSim nutná k vygenerování tohoto počtu hodnot byla na mém počítači (1.5GHz) přibližně 30 hodin. Simulace byla ale nakonec úspěšně provedena. Výsledky testů byli pozitivní (vynikající). Konkrétní dosažené hodnoty jsou přiloženy v souboru fibresults.txt.

4.Závěr

Implementovaný lineární kongruentní generátor je možné aplikovat na stejných místech, jako softwarový generátor. Je možné jej nastavit na stejné konfigurace parametrů. Kvality generované posloupnosti závisí na konkrétním nastavení.
Volba mezi softwarovým a hardwarovým řešením generátoru už závisí na prioritních požadovaných vlastnostech. Hardwarová realizace by umožňovala rychlejší generování, ale nemusela by se cenově vyplatit.
Tyto generátory se obecně užívají spíše při potřebě obstarat „relativně náhodná“ čísla s rovnoměrným rozložením. Pro aplikace, ve kterých je požadována nízká korelace mezi jednotlivými hodnotami, jako je například Šifrování a kryptografie není tento generátor v základní podobě vhodný. V tomto oboru se užívají často generátory LFSR, které by ale byli také v základní podobě nedostatečné a proto se používají v zapojeních, ve kterých se kombinuje několik generátorů, přičemž výstup některého může zpětnou vazbou ovlivňovat zbývající (použití výstupu jako hodinového signálu) nebo jsou výstupy ještě nějak modifikovány. Podobné experimenty by bylo samozřejmě možné provádět i s LCG a pravděpodobně by bylo možné s jejich pomocí také vytvořit vhodný generátor. LFSR se ale v kryptografii užívají pro generování bitových posloupností, které se využívají při proudovém šifrování a LCG generují v každém taktu k-tice bitů, jejichž šířka závisí na modulu generátoru. Pro využití v proudovém šifrování by tedy bylo ještě nutné upravit výstup tak, aby generoval jednotlivé bity. Aplikace LFSR je tedy v této oblasti asi jednodušší.
V praxi se LCG užívají tak, že se po několika krocích změní parametry a, c. Jestli by byl takový způsob užití dostatečný pro využití v šifrování nedokážu posoudit.

Fibonacciho (lagged) generátor s úpravou podle Lüschera vykazuje v testech výrazně lepší výsledky než LCG. I v praxi je užíván v základní verzi již řadu let. Generuje opravdu kvalitní pseudohodné hodnoty s v porovnání s LCG dlouhou periodou. Korelace mezi hodnotami u něj v podstatě nenastává. Uplatnění může najít při jakékoli potřebě pseudonáhodných čísel. Domnívám se, že i v kryptografii. Generátory se užívají jen v několika konfiguracích v závislosti na konstantách c_1 a c_2, ale jimi generovaná posloupnost závisí řádově na stovkách (nebo tisících) náhodných čísel, v relativně velkém (a snadno rozšiřitelném) bitovém rozsahu. Pro aplikaci při proudovém šifrování by se musel modifikovat výstupní posuvný registr. Byl by složen z jednobitových registrů a musel by pracovat na výrazně vyšší taktovací frekvenci.

Namodelované hardwarové generátory mají oproti softwarovým výhodu v tom, že je možné je jednoduše modifikovat pro velké hodnoty modulů. Neexistuje zde jiné omezení než cena součástek, kdežto softwarové generátory jsou omezeny 32-bitovým, popř. 64-bitovým rozsahem hodnot v závislosti na architektuře počítače, na kterém jsou provozovány.
V současnosti se pro generování pseudonáhodných čísel nejvyšší kvality užívají kombinace generátorů, pracující na odlišných principech a jejich moduly jsou často větší (i několikanásobně) než moduly mnou implementovaných generátorů.
S generátory by jistě bylo možné ještě dále experimenrovat. Mohli bychom např. realizovat kombinovaný generátor. Kdybychom realizovali kombinační obvod LCG jako řetězený, měli by oba generátory přibližně stejné zpoždění a mohli by generovat hodnoty současně, aniž by musel jeden z nich čekat. Jimi generované hodnoty by bylo možné poté libovolně vzájemně modifikovat. Inicializace LCG by se mohla provést za pomoci vstupního posuvného registru Fibonacciho generátoru atd.

5. Použitá literatura a zdroje informací.

[1] Donald E. Knuth : The Art of Computer Programming, volume 2 - Seminumerical Algorithms, Third edition, 1998, Addison-Wesley.
[2] Pavel Holan : Generátory náhodných čísel, 1989, JZD Agrokombinát Slušovice.
[3] Vladimír Drábek : Spolehlivost a diagnostika, 2. vydání, 1986, Brno VUT.

Zbytková aritmetika:
http://whyslopes.com/Number_Theory/Decimals_Modular_and_Remainder_Arithmetic.html

Přehled LCG generátorů: http://random.mat.sbg.ac.at/~charly/server/node3.html

Obecné informace o generátorech náhodných čísel :
http://www.stahroun.me.cz/vypstat/numerika/generovani/

Stránky se softwarem pro testování pseudonáhodných posloupností :
http://www.stat.fsu.edu/pub/diehard/

Testování náhodných čísel : http://random.mat.sbg.ac.at/tests/

J. Hrubý, L. Andrej : Kvantový šumátor a jeho testování (technická zpráva Akademie Věd České Republiky)
http://www.cs.cas.cz/ftp://ftp.cs.cas.cz/pub/reports/v948-05.pdf

Vlastimil Martinek : Aplikace simulačních metod ve spolehlivosti (dokument ústavu automatizace a informatiky, FSI VUT v Brně).
http://www.fs.vsb.cz/akce/2001/ASR2001/Proceedings/papers/40.pdf

6.Přílohy

6.1 Testování generátorů

6.1.1 Úvod

Při řešení projektu se bylo nutné seznámit i se způsobem testování PRNG. Tato činnost vyplnila také jistou nezanedbatelnou část mého času. Přikládám jsem proto text pojednávající o této problematice, aby tento čas nezůstal nevyužit.
V této příloze je popsána problematika testování PRNG a více přiblíženy jsou nejčastěji používané testy.
Informace uvedené v této kapitole nejsou stěžejní pro porozumění návrhu generátoru, ale jsou zde spíše pro úplnost. Na konkrétní zde uvedené informace, které byly potřebné pro porozumění textu, jsou v příslušnýh kapitolách odkazy.

Požadavky na generované posloupnosti

K základním požadavkům ze statistického hledika patří:

• Rovnoměrné pokrytí celé množiny hodnot
• Nízká korelace mezi prvky posloupnosti
• Zajímá nás *stacionarita* generované posloupnosti, zejména zjišťujeme existenci případného nenulového trendu. Není třeba zabývat se existencí trendu příliš podrobně, neboť jeho přítomnost v testované posloupnosti lze odhalit nejsnáze, a navíc je přítomnost trendu známkou jen skutečně nekvalitního generátoru.
• Dále se zabýváme *periodicitou*, tedy existencí významných cyklů v generované posloupnosti.

Dalšími nestatistickými požadavky jsou zřejmě maximální rychlost generování a maximální perioda generátoru.
V praxi (např. v oblasti kryptografie) se generátory testují některou baterií statistických testů, které jsou specifikované i s požadovanými výsledky v příslušných normách.

Dělení testů

V současnosti již existuje široká škála testů, které jsou zaměřené na ověření různých vlastností náhodných posloupností. Při jejich návrhu a při testování psedonáhodných posloupností se užívá zákonů teorie statistiky a pravděpodobnosti.

Testy PRNG se dělí do dvou základních skupin:

1) Teoretické testy
2) Empirické testy

6.1.2 Empirické testy

Empirické testy pracují přímo s hodnotami generovaných posloupností, kterých musí být dostatečný počet. Přistupují ke generátorům jako k černým skříňkám, o jejichž funkčnosti se nic neví. Pro ověření skutečnosti, že posloupnost čísel splňuje určité rozložení pravděpodobnosti se standardně užívají testy:

- Chí kvadrát test (test dobré shody)
- Kolmogorov-Smirnov test (K-S test)

Základní testy

Chí-kvadrát test (test dobré shody)

Sestaví se histogram s počtem intervalů k z dostatečně velkého počtu prvků n vygenerované posloupnosti. Symbolem p_s označíme pravděpodobnost, že výsledek pokusu padne do intervalu s a y_s nechť je počet pokusů, které skutečně padly do kategorie s. Poté můžeme zformulovat vztah

$$D = \Sigma \ ((y_s - np_s)^2 / np_s), \quad \text{kde } s = 1,..k$$

Jsou-li data přímo shodné s požadovaným rovnoměrným rozdělením, tak je D = 0. Tento případ ve skutečnosti skoro nenastává. D má rozdělení chí-kvadrát s k-1 stupni volnosti. Nulová hypotéza o tom, že vygenerované čísla pochází z rovnoměrného rozdělení, nemůže být zamítnuta na hladině významnosti *a* , jestliže vypočítané D je menší, než $\chi 2[1-\alpha; k-1]$. Hladinu významnosti můžeme určit podle statistických tabulek, jestliže známe počet stupňů volnosti a výsledek testu χ_2. Chí-kvadrát test byl navržen pro diskrétní rozdělení a pro velký rozsah dat. Pro spojité rozdělení je pouze přibližný.

Kolmogorov-Smirnov test (K-S test)

Tento test umožňuje určit, jestli n náhodně vygenerovaných čísel je ze specifického souvislého rozdělení. Je speciálně zhotoven pouze pro malé příklady a spojité rozdělení a jedná se tedy o analogii Chí-kvadrát testu pro spojité rozdělení. Výpočet je naprosto odlišný, ale protože tento test nebude dále v textu vůbec figurovat, nebudu jej uvádět.

Dva právě uvedené testy je možné považovat za základní a užívají se jako součást jiných komplikovanějších testů. Uvedeme zde jen ty nejznámnější.

Testy rovnoměrnosti rozložení

k-dimensional uniformity (k-distributivita).

Tento pojem je nutné ozřejmit pro testy rovnoměrnosti rozložení pro vyšší dimenze než je jedna (Serial test, Spectral test).

Předpokládejme, že u_n je n-té číslo v sekvenci náhodných čísel rovnoměrně rozdělených v intervalu R<0, 1). Nechť čísla a_1 a b_1 jsou také z tohoto intervalu, kromě toho platí $b_1 > a_1$. Potom pravděpodobnost P, že u_n leží v intervalu $<a_1, b_1)$ je rovna :

$$P(a_1 \le u_n < b_1) = b_1 - a_1 \quad \forall b_1 > a_1$$

Toto je označováno jako vlastnost 1-distributivity pro u_n.

2-distributivita je generalizací této vlastnosti v dvojdimenzionálním prostoru. Pro výpočet je potřeba pár po sobě jdoucích hodnot u_{n-1} a u_n.

Dostaneme následující výraz:

$$P((a_1 \le u_{n=1} < b_1) \wedge (a_2 \le u_n < b_2)) = (b_1 - a_1)(b_2 - a_2)$$

Zároveň musí platit, že a_1, b_1, a_2, b_2 jsou z intervalu <0,1), a platí $b_1 > a_1$ a $b_2 > a_2$.

Posloupnost se nazývá k-distributivní, jestliže:

$$P((a_1 \le u_{n-1} < b_1),\ldots,(a_k \le u_{n+k-1} < b_k)) = (b_1 - a_1)\ldots(b_k - a_k)$$

Pro všechny možnosti musí platit: a_i, b_i z intervalu <0, 1) a $b_i > a_i$ pro i = 1,2,….,k.

Platí, že k-distributivita je vždy i k-1 distributivitou, ale ne naopak.

Test rovnoměrnosti rozložení (v 1D, Frequency test).

Tento test je možné realizovat buď Chí- kvadrát testem, kde $p_s = 1/k$ nebo K-S testem.

Test mezer

Uvažujeme-li tři sousední čísla posloupnosti, pak existuje právě 6 možností vzájemných relací: a<b<c, a>b>c, a<c<b, a>c>b, c<a<b, c>a>b. Možnost, že by se dvě nebo tři čísla v trojici rovnala, vylučujeme. Při vhodně zvolených konstantách generátoru se to nestává ani u pseudonáhodného generátoru. Zjistíme, kolikrát se ve zkoumané posloupnosti objeví každá z těchto možností a statisticky je zhodnotíme testem $\chi 2$ s tím, že předpokládáme, že pravděpodobnost je vždy rovna jedné šestině a počet stupňů volnosti pro Chí-kvadrát je 6.

Two-level test

Testy, které byly uváděny doposud mají jeden podstatný problém. Jestliže je velikost vzorku malá, výsledky se mohou aplikovat lokálně, ale ne globálně na celý cyklus. Podobně, výsledky globálního testu se nemusí dát aplikovat lokálně a proto zde může být značná závislost v části posloupnosti. Pro překonání tohoto problému se používá two-level test. Existují dva typy tohoto testu:

- Chí-kvadrát na Chí-kvadrát test
- K-S na K-S test

Principem obou typů je, že se datová vzorka rozdělí na n intervalů o velikosti k a provede se n testů jednoho typu, jejichž výsledky se použijí jako vstup do dalšího testu, stejného typu.

Serial test

Tento typ testu se používá pro testování rovnoměrnosti ve dvou dimenzích a výš, jedná se tedy o určení již předeslané k-distributivity. Pro výpočet ve dvou dimenzích se rozdělí prostor intervalu R<0, 1) na k rovnoměrných intervalů. Z náhodně vygenerovaných čísel $\{x_1, x_2,\ldots x_n\}$ z intervalu R<0, 1) vytvoříme n/2 nepřekrývajících se párů (x_1, x_2), (x_3, x_4),….. . Z těchto dvojic vytvoříme body, na osu x nakreslíme první z dvojice čísel a na osu y druhé číslo. Každý z těchto bodů padne do nějakého z K^2 čtverců. V ideálním případě do každého čtverce padne $n/(2K^2)$ bodů. Pro nalezení odchylky aktuálních bodů od očekávaných se použije Chí-kvadrát test se stupněm volnosti $K^2 – 1$ (jednotkový čtverec se v praxi dělí zpravidla na 100 dílů a v jedné serii se zkoumá asi 50000 hodnot). Při testu rovnoměrnosti trojic postupujeme stejně jako v předcházejícím, bereme však trojice náhodných čísel a uvažujeme jednotkovou krychli. Pro vyšší dimenze se postupuje analogicky.
Závislost po sobě jdoucích čísel v posloupnosti se projeví jako nerovnoměrnost ve vyšší dimenzi.

Rozložení maxima z *n* členů

V tomto testu bereme n-tice po sobě následujících náhodných čísel $x_1, x_2, x_3,\ldots , x_n$. Vypočítáme hodnoty $U=[max(x_1, x_2, x_3,.. , x_n)]_n$. Takto vzniklou posloupnost hodnot U testujeme na rovnoměrnost (volíme $n=2, 3, 4, 5, 10$ a v jednom pokusu testujeme 100 tisíc hodnot).

Hamingův test

Tento test již nespadá do třídy empirických testu, které využívají kritéria χ^2. Test má odhalit, zda se některé hodnoty náhodných čísel nevyskytují s větší četností. Zkoumá se velikost součtu:

$$\sum_{i=1}^{n-1}(y_i - 0.5)(y_{i+1} - 0.5) \to 0$$

kde *n* je počet vygenerovaných čísel.
Je-li generátor dobře navržen, je hodnota tohoto součtu přibližně nulová. Zkoumané hodnoty musí být při užití tohoto vztahu z intervalu R<0, 1), ověřuje se normalizované rovnoměrné rozložení.

Testy náhodnosti rozložení

Test na rovnoměrnost by mohl dostat dobré výsledky i v případě posloupnosti čísel, která by byla nedostatečně náhodná. Příkladem takové posloupnosti je např.

111122223333444455 55666.

Je zřejmé, že počet členů posloupnosti, které padnou do jednotlivých intervalů sice odpovídá rovnoměrnému rozložení, ale tuto posloupnost určitě nemůžeme nazvat náhodnou. Proto zavádíme testy náhodnosti rozložení.

Serial – correlation test

Tato metoda je založena na testování závislosti dvou náhodných čísel. Dvě čísla jsou na sobě závislá, je-li jejich kovariance nenulová. Ale i v případě, že je kovariance rovna nule, můžou být čísla na sobě závislá. Máme-li sekvenci náhodně vygenerovaných čísel, můžeme vypočítat kovarianci mezi čísli vzdálenými od sebe o k, tj. mezi x_n a x_{n+k}. To se nazývá autokovariance se zpožděním k. Označíme-li ji jako R_k, můžeme napsat

$$R_k = (1 / (n-k)) \sum_{i=1}^{n-k} (U_i - (1/2))(U_{i+k} - (1/2)),$$ kde Ui jsou pseudonáhodné hodnoty

z intervalu R<0,1).

R_k leží vždy v intervalu R<-1, 1>. Čím blíže leží hodnoty nule, tím jsou hodnoty dvou čísel nezávislejší. Rovnost $R_k = 1$ nebo $R_k = -1$ naopak indikuje totální lineární závislost. Dobrá hodnota R_k by měla ležet mezi hodnotami $\mu_n - 2\sigma_n$ a $\mu_n + 2\sigma_n$, kde $\mu_n = -1/(n-1)$,

$$\sigma_n^{\ 2} = \frac{n^2}{(n-1)^2(n-2)}, n > 2.$$

Rk by mělo ležet mezi těmito limity přibližně v 95% času.

Pro velké n je R_k normálním rozdělením s nulovou střední hodnotou a rozptylem 1/[144(n – k)]. Pro 100(1 - α)% interval spolehlivosti platí

$$R_k \pm (z_{1-\alpha/2} / (12\sqrt{n - k})).$$

Jestliže tento interval neobsahuje nulu, potom můžeme říct, že sekvence má podstatnou korelaci (significant correlation).
Předchozí vztahy ovšem platí pouze pro k ≥ 1. Pro k = 0 se počítaná hodnota R_0 očekává $R_0 = 1/12$ pro čísla z intervalu R<0,1).

Test na intervaly

V tomto testu prověřujeme délku posloupnosti náhodných čísel mezi dvěma hodnotami, které padnou do téhož předem zvoleného intervalu. Vyčíslíme tedy počty posloupností stejných délek a statisticky je zhodnotíme kritériem Chí-kvadrát.
Zavedeme-li hodnotu *p = horní mez intervalu – dolní mez* intervalu, pak pravděpodobnost, že posloupnost bude mít délku *t*, je:

$$p_t = p(1-p)^t,$$

tedy pro posloupnost délky nula platí $p_0 = p$. Tohoto testu se dá využít pouze pro generátor čísel typu *real* s rozložením R<0,1).
Stupeň volnosti Chí-hvadrát pro tento test volíme *e / (horní mez intervalu–dolní mez)*, přičemž *e* je základ přirozeného logaritmu. Toto tvrzení plyne ze skutečnosti, že pokud *p* je šířka intervalu, pak *p* je taky pravděpodobnost že hned další číslo padne do téhož intervalu. Nejsnažší cestou je, že *1/p*-té číslo padne do téhož intervalu, pravděpodobnost tohoto tvrzení je rovna $p_t = p(1-p)^{(1/p)}$, přičemž $t = 1/p$. Výraz $(1-p)^{(1/p)}$ *se* blíží hodnotě *1/e*. Z tohoto tvrzení plyne, že výsledek výrazu p_t se vždy blíží hodnotě *p/e*. A odtud *df* (stupeň volnosti) = *e/p*.

Test zvaný RunUp

V tomto testu sledujeme vždy neklesající sekvence čísel a počítáme četnosti jednotlivých délek těchto neklesajících posloupností. Četnosti počítáme pro délky jedna až pět a všechny delší zařazujeme do kolonky s délkou šest. Vezměme si pro příklad označení těchto četností r_i, kde $i = 0..6$. Výpočet χ^2 lze provést na základě vzorce

$$\chi^2 = \frac{1}{n}\sum_{i=1}^{6}\sum_{j=1}^{6} a_{ij}(r_i - n\cdot b_i)(r_j - n\cdot b_j)$$

kde *n* je počet vygenerovaných čísel, a_{ij} a b_i jsou následující konstantní matice:

a_{ij} = {{4529.4, 9044.9, 13568, 18091, 22615, 27892}, {9044.9, 18097, 27139, 36187, 45234, 55789}, {13568, 27139, 40721, 54281, 67852, 83685}, {18091, 36187, 54281, 72414, 90470, 111580}, {22615, 45234, 67852, 90470, 113262, 139476}, {27892, 55789, 83685, 11580, 139476, 172860}}

b_i = {1.0/6.0, 5.0/24.0, 11.0/120.0, 19.0/720.0, 29.0/5040.0, 1.0/840.0}

Počet stupňů volnosti pro χ^2 test je roven 6.

Poker test

Posloupnost generovaných čísel rozdělíme do pětic, pak zjistíme počet různých čísel v pětici, který označíme např. písmenem *r*. Poté použijeme následujícího kritéria s pravděpodobnostmi

$$p_r = \frac{d(d-1)..(d-r+1)}{d^k} \left\{ {k \atop r} \right\}$$

kde p_r je pravděpodobnost, že v pětici bude právě r různých čísel, k je počet čísel ve skupině (v našem případě 5, počítáme pětice), d je maximální možná hodnota generované veličiny zvětšené o 1, $\left\{ {k \atop r} \right\}$ je Strilingovo číslo druhého druhu. Hodnoty pravděpodobnosti p_r se užijí jako hodnoty teoretické pravděpodobnosti v Chí-kvadrát test. Tohoto testu se dá využít pouze pro testování posloupností, jejichž členy jsou čísla typu integer a padnou do intervalu <0, d -1>.

STIRLING NUMBERS S(p, k) OF THE SECOND KIND

	k=	0	1	2	3	4	5	6	7	8	9	10
p=												
0		1										
1		0	1									
2		0	1	1								
3		0	1	3	1							
4		0	1	7	6	1						
5		0	1	15	25	10	1					
6		0	1	31	90	65	15	1				
7		0	1	63	301	350	140	21	1			
8		0	1	127	966	1701	1050	266	28	1		
9		0	1	255	3025	7770	6951	2646	462	36	1	
10		0	1	511	9330	34105	42525	22827	5880	750	45	1

Tabulka 0-6 – tabulka Stirlingových čísel druhého druhu

Test sběratele kupónů (Coupon collector's test)

Zjišťuje se délka posloupnosti náhodných čísel taková, že se v ní každý možný člen posloupnosti objeví alespoň jednou. Pro kritérium Chí-kvadtrát se používá vztahu pro q_r , kde q_r je pravděpodobnost, že posloupnost délky r neobsahuje všechna čísla generovaná generátorem. Ve vztahu figuruje největší možné číslo, které generátor může vytvořit d a $\left\{ {r \atop d} \right\}$, což je Stirlingovo číslo druhého druhu. Test sběratele kupónů se dá použít pro generátor čísel typu integer v intervalu <0, d-1>.

Collision test

Mějme m urn a n koulí, které vrhneme náhodně do těchto uren, přičemž m je mnohem větší než n. Pravděpodobnost, že jedna koule spadne do určité urny je pro všechny koule stejná. Většina koulí při vrhu přistane v prázdných urnách. Jestliže koule spadne do urny, která již obsahuje nějakou kouli, nastává „kolize". Pravděpodobnost, že urna bude po vrhu obsahovat k koulí je

$p_k = \left\{ {n \atop k} \right\} m^{-k} (1 - m^{-1})^{n-k}$, takže očekávaný počet kolizí na urnu je

$$\sum_{k\geq 1}(k-1)p_k = \sum_{k\geq 0} kp_k - \sum_{k\geq 1} p_k = n/m - 1 + p_{0.}$$

Pro pravděpodobnost, že v urně nastane c kolizí platí vztah $\frac{m(m-1)...(m-n+c+1)}{m^n}\left\{ {n \atop n-c} \right\}$.

<u>Birthday spacings test</u>

Máme m uren a n koulí stejně jako v Collision testu, ale urny nyní představují dny roku a koulím se po dopadu do urny přiřadí její hodnota. Tato hodnota reprezentuje den narozenin. Předpokládejme, že narozeniny jsou (Y_1,…,Y_n), kde $0 \leq Y_k < m$. Seřaďme je do neklesajícího pořadí $Y(1) \leq ... \leq Y(n)$, potom můžeme definovat S_1=Y(2)-Y(1), …, S_{n-1} = Y(n)-Y(n-1), S_n = Y(1) + m – Y(n).
Nakonec vzestupně seřadíme rozdíly $S(1) \leq ... \leq S(n)$. Spočítáme teoretickou pravděpodobnost pro různé počty rovnajících se rozdílů nebo-li počet indexů j, pro které platí $1 < j \leq n$ a S(j) = S(j-1). Empiricky se test provede přibližně 1000 krát a zjištěné výsledky se porovnají s teoretickými pomocí testu dobré shody se třemi stupni volnosti.

Uvedené testy a ještě i některé další, jako třeba Permutation test jsou podrobněji uvedeny např. v [1] .

D.Knuth (v [1]) provádí i zhodnocení testů. Uvádí, že jako „nejslabší se zdají být“ frequency test a serial correlation test v tom smyslu, že většina generátorů jimi úspěšně projde, resp. neprojde jimi jen skutečně nekvalitní generátor. Naopak doporučeny jsou Run test a Collision test.

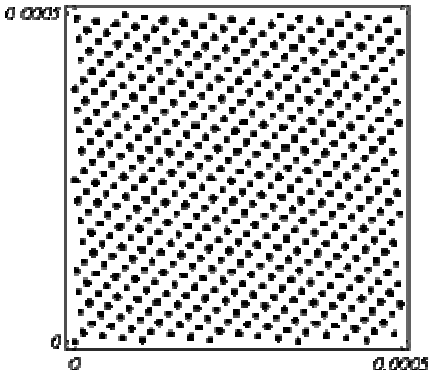
6.1.3 Teoretické testy

Teoretické testy nevyužívají samotných hodnot vygenerovaných posloupností, ale zaměřují se na určení charakteristik posloupností využívaje číselné teoretické metody založené na rekurentním pravidle použitém pro vytvoření posloupnosti a zkoumají vliv jednotlivých parametrů generátoru na vygenerovanou posloupnost. Nejsou definovány pro všechny druhy generátorů. Tyto testy se často užívají jako předpověď úspěšnosti generátoru v empirických testech, ale nemůžou dát záruku úspěšnosti v numerické praxi.
Do této kategorie spadá například pozorování vygenerované posloupnosti ve formě několika-bitových čísel. Pouhým okem je pak možné v grafickém provedení vypozorovat případné korelace (shody) v náhodné posloupnosti, které by neodhalil žádný z testů. Dalším testem může být i pozorování četnosti vygenerované posloupnosti, která by měla být téměř pro všechny vzorky (několika-bitová čísla) obdobná.
Mezi další možnosti teoretických testů může např. patřit komprese vygenerované bitové posloupnosti. Kompresní techniky využívají zejména korelace. Pokud tedy po zabalení posloupnosti bude mít soubor tutéž velikost, je generátor pravděpodobně bez korelací.

Spectral test

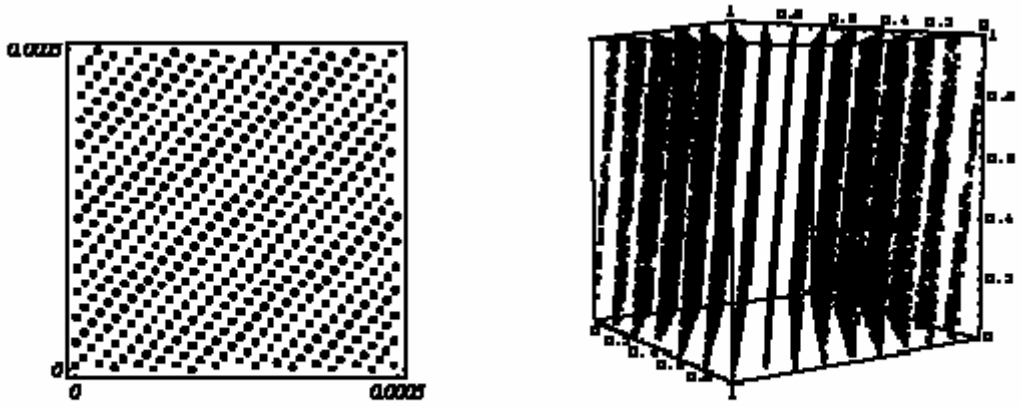
Tento test se považuje za nejvíce směrodatný, co se týká rovnoměrnosti rozložení u lineárních kongruentních generátorů. Jedná se o další z testú pro určení k-distributivity (první byl serial test). Tento test podává informaci o tom, jak hustě k-tice $\{x_1, x_2,\ldots x_k\}$ vyplní k dimenzionální prostor. U pseudo náhodných generátorů nejsou body v mřížových plochách rozmístěny náhodně kdekoli v prostoru, ale jen na určitém počtu nadrovin (nadrovinou zde míníme prostor o jednu dimenzi nižší než výchozí prostor, do kterého umísťujeme k-tice). Spektrální test rozhodne maximální odchylku mezi sousedními nadrovinami. Čím je větší odchylka, tím je generátor horší. Situace je dobře zobrazitelná pro k = 2 a k = 3.

SIMSCRIPT



Obr.1 Výsledná mřížová plocha spectral testu pro výše uvedený generátor.

Vzorec z bodů na obr.1 je možné pokrýt určitým počtem rovnoběžných přímek. Existuje ale více odlišných skupin rovnoběžných přímek, které projdou všemi body. Např. přímky svírající úhel 45° s vodorovnou osou, přímky svírající úhel cca 135° nebo téměř vodorovné přímky svírající úhel přibližně 160° (úhly jsou počítány vzhledem k pomyslnému směru osy x). V trojrozměrném prostoru se bude jednat o rovnoběžné roviny a ve vyšších rozměrech o rovnoběžné nadroviny (hyperplanes).



Př.3- ANSI LCG (2^31, 1103515245, 12345, 12345)

Př.2- RANDU LCG (2^31, 2^16+3=65539, 0, 1)

Obr.2 Mřížové plochy pro některé skutečné pdeudonáhodné generátory.

Takovéto systematické uspořádání patrné z testů může „vypadat tak nenáhodně“, že může svádět k tendenci zavrhnutí kongruentních generátorů. Je ale zapotřebí si uvědomit, že zde hraje zásadní roli přesnost generovaných hodnot. Budeme-li mít skutečně náhodná čísla v intervalu R<0, 1) a zaokrouhlíme je na konečnou přesnost tak, že každá hodnota bude násobkem 1/v pro dané v, potom body v t-rozměrném prostoru budou pravidelně uspořádány (ve 2D vznikne mřížka). Nechť je $1/v_2$ maximální vzdálenost mezi čarami ze všech skupin rovnoběžných rovných čar, které pokrývají body {(x/m,s(x)/m)} ve dvou dimenzích, v_2 nazýváme dvou rozměrnou přesností generátoru náhodných čísel. Je zřejmé, že hustota k-tic se bude se zvětšujícím rozměrem snižovat. Může ale nastat případ (a často nastává), že $v_{k1} > v_{k2}$ pro $k_2 > k_1$, protože v_k je závislé na konkrétním vzájemném uspořádání nadrovin v k-dimenzionálním prostoru. Větší hodnota modulu m znamená větší přesnost, jestliže hodnoty normalizujeme do intervalu <0,1). Se zvětšujícím se m se zvětší počet možných nadrovin (souvislost s potency) a to vede ke zmenšení absolutní vzdálenosti mezi nadrovinami v jednotkovém tělěse, pokud zvolíme vhodně všechny parametry.

Pro zhodnocení testování je užitečné mít hodnotu, která umožní zjistit, jak kvalitní je rozložení vzhledem k nejlepšímu možnému pro daný modul m při použití multiplikativního koeficientu a a aditivní konstanty c. Tyto hodnoty mohou být označeny různě a většinou se výsledky spectral testů uvádějí právě v nich. Označme je např. písmenem s. Hodnoty s_k samozřejmě závisí přímo na hodnotách v_k, ale jejich matematické vztahy pro různé k se liší. Jestliže jsou normalizované, vycházejí v intervalu <0,1). Dosažení hodnoty $s_k = 1$ znamená dosažení maximální možné teoretické hodnoty v_k pro dané k a m. Čím blíže je hodnota jedné, tím lepší konstanty a a c jsme pro daný modul m použili. Rozměr k se v praxi používá k = 2 , …, 8. Veličinou indikující skutečný stupeň přesnosti náhodnosi je v_k (vysoké hodnoty sk jsou becenné při malé hodnotě m). Počet bitů potřebných pro binární reprezentaci hodnot s p řesností v_k je $\log(v_k)$.

		$LCG(2^n, a, 1, 0)$						
n	a	$s = 2$	3	4	5	6	7	8
15	13821	0.839	0.302	0.673	0.703	0.698	0.476	0.545
16	47485	0.79	0.511	0.541	0.839	0.546	0.61	0.661
17	82669	0.915	0.7	0.609	0.308	0.435	0.552	0.648
18	160461	0.921	0.375	0.792	0.464	0.671	0.771	0.665
19	217293	0.921	0.298	0.777	0.665	0.742	0.716	0.61
20	387645	0.837	0.716	0.529	0.685	0.67	0.665	0.53
21	886269	0.899	0.605	0.476	0.653	0.685	0.557	0.688
22	3067221	0.9	**0.06**	0.2	0.414	0.579	0.798	0.713
23	3513381	0.9	**0.06**	0.2	0.414	0.579	0.798	0.713
24	12268885	0.896	0.68	0.551	0.332	0.421	0.59	0.695
25	21149085	0.847	**0.038**	0.142	0.314	0.52	tabular335	0.707
26	41430805	0.804	0.634	0.78	0.538	0.637	0.599	0.607
27	96801245	0.861	0.81	0.533	0.818	0.679	0.64	0.604
28	169764749	0.825	0.89	0.713	0.694	0.577	0.767	0.586
29	338335805	0.804	0.68	0.709	0.736	0.509	0.561	0.656
30	777138309	0.705	0.849	0.597	0.457	0.805	0.607	0.688
31	906185749	0.728	0.625	0.457	0.488	0.553	0.684	0.608
32	3039177861	0.946	0.612	0.42	0.631	0.525	0.493	0.643
33	6223592213	0.865	0.447	0.59	0.69	0.697	0.734	0.637
34	10618587253	0.92	0.719	0.595	0.46	0.405	0.583	0.582
35	21733031525	0.754	0.389	0.912	0.76	0.594	0.54	0.592
		0.771	0.502	0.77	0.58	0.524	0.614	0.6

Tab.2 Tabulka spektrálních testů některých LCG generátorů.

Důležitým faktem mluvícím pro užití spektrálního testu je i skutečnost, že serial testem v několika dimenzích projde každý generátor, který projde spectral testem. Pokud tedy provádíme spectral test, je bezpředmětné provádět serial test. Tato skutečnost plyne ze studií Haralda Niderreitera a důkaz je také v [1].