Zufallszahlengeneratoren mit zentralem Grenzwertsatz und Lötkolben

Rolf Freitag

21C3, Dezember 2004

Inhaltsverzeichnis

1 Einführung

Es gibt zwei Arten von Zufallszahlengeneratoren:

- Pseudozufallszahlengeneratoren, die Zufallszahlen berechnen und auch als eine endliche Folge von Zahlen dargestellt werden können, also eine endliche Periode haben und leicht in Software realisiert werden können.
- Echte Zufallszahlengeneratoren, die physikalische Zufallsprozesse in Zufallszahlen umsetzen und daher eine unendliche Periodenlänge haben, also Hardware mit Rauschen benötigen.

Zudem gibt es Mischformen, beispielsweise ein Pseudozufallszahlengenerator, der bei jedem oder seinem ersten Aufruf mit der Systemzeit (time(NULL)) initialisiert wird.

“Is there a need for True Random Numbers - produced by a Hardware Generator?

There is a DEFINITE NEED FOR DATA WITH ENTROPY NOT BASED ON ALGORITHMS. As to whether we need *true* random numbers, that’s a subject for debate. Typically we get data that’s as good as we need it to be from a hardware generator by running the initial data (which may have some kind of pattern or bias) through an algorithm which sort of mixes it. But it is very usefull to have this entropy come from a hardware device, because it affords an EXTRA LEVEL OF SECURITY; pseudorandom numbers can be compromised if the algorithm is known and the seed is discovered, but THERE IS NO SUCH EQUIVALENT FOR RANDOM NUMBERS GENERATED BY A HARDWARE DEVICE.´´

Moses Liskov (faq-editor@rsa.com) http://www.rsa.com/ year 2000

Neben der Entropie ist ein anderer wichtiger und verwandter Aspekt die Qualität von Zufallszahlengeneratoren, zu der in den C-FAQ und den Numerical Recipes einiges steht:

“If all scientific paper whose results are in doubt because of bad rand()s were to disappear from library shelves, there would be a gap on each shelf about as big as your fist.´´

Numerical Recipes (in C), Chapter 7.1

Bei den Pseudozufallszahlengeneratoren muß deshalb immer ein Kompromiß zwischen Geschwindigkeit und Qualität gefunden werden.
Bei den echten Zufallszahlengeneratoren hingegen gibt es das Problem, dass sie im Vergleich zu Pseudozufallszahlengeneratoren meist sehr langsam sind, wie man z. B. an /dev/random sieht.

2 Theorie

Modulo-2-Summe von zwei Zufallsbits

Die Entropie einer Zufallsbitsequenz

$$E = -p(0) \cdot log_2(p(0)) - p(1) \cdot log_2(p(1)) \tag{1}$$

ergibt sich aus der Wahrscheinlichkeit $p(0)$ (abgek. x), dass ein Bit der Sequenz 0 ist und aus der Wahrscheinlichkeit $p(1)$ (abgek. 1-x), dass ein Bit der Sequenz 1 ist.

Damit ist es einfach, den Zuwachs an Entropie je Bit zu berechnen, den man erhält, wenn man eine Bitfolge mit einer anderen, statistisch unabhängigen mit gleichen Wahrscheinlichkeiten ($p(0)$ und $p(1)$) exklusiv verodert und so eine sekundäre Bitfolge durch bitweise Modulo-2-Addition bildet:

$$\begin{aligned} E_1 - E_2 = &-(x^2 + (1-x)^2) \cdot log_2(x^2 + (1-x)^2) \quad -2 \cdot x \cdot (1-x) \cdot log_2(2 \cdot x \cdot (1-x)) \\ &+x \cdot log_2(x) + (1-x) \cdot log_2(1-x) \end{aligned} \tag{2}$$

mit den Abkürzungen $x = p(0)$ und $1 - x = p(1)$.

Für nicht perfekt zufällige primäre echte Zufallsbitsequenzen, also $0 < x < 1, \ \ x \neq 0.5$, ist dieser Zuwachs positiv und größer null, wie sich auch beim Auftragen des Zuwachses $y = E_1 - E_2$ über $x = p(0)$ zeigt: (Abb. 1) Durch rekursive Anwendung ergibt sich hieraus, dass die Modulo-2-Addition von 2^n ($n > 0$) statistisch unabhängigen Bitsequenzen mit gleichen Wahrscheinlichkeiten eine entropiereichere sekundäre Bitsequenz ergibt, die umso entropiereicher (=zufälliger) ist, je größer n ist.

Weil die Gleichung 2 unabhängig von den einzelnen Bits gilt, werden durch die Modulo-2-Addition auch die Korrelationen innerhalb der sekundären Bitsequenz abgeschwächt, sodass auch der Entropiebelag erhöht wird. Wie sich numerisch leicht zeigen lässt, ist das Voraussetzen von statistisch unabhängigen Bitsequenzen mit gleichen Wahrscheinlichkeiten in der Regel nicht nötig. Dadurch ist es in der Praxis fast immer möglich, mittels Modulo-2-Addition von 2^n primären Zufallsbitfolgen eine Zufallsbitfolge mit erhöhter Entropie zu erzeugen.

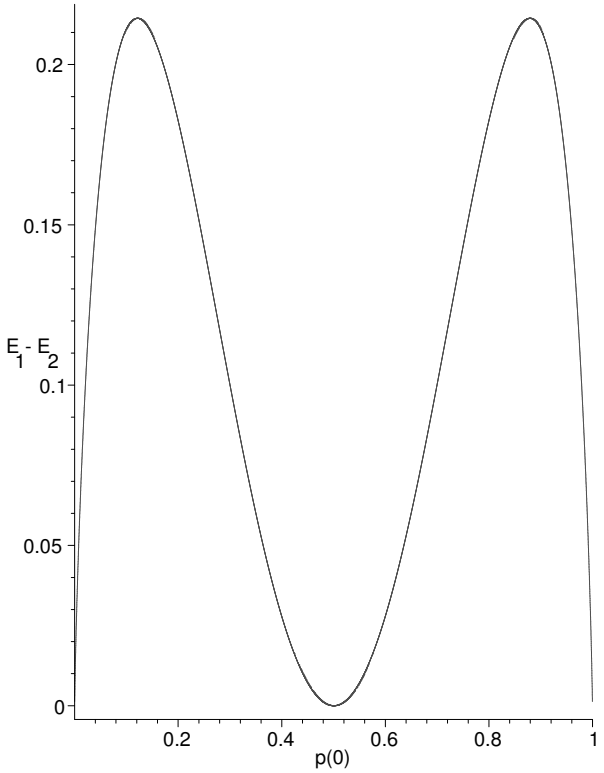


Abbildung 1: Entropiezuwachs durch Exklusiv-Oder-Verknüpfung von zwei Zufallsbits, aufgetragen über p(0) (abgek. x).

Beispiel: Zwei unabhängige Bit-Folgen, die beide zu 2/3 aus 0 und 1/3 aus 1 bestehen (z. B. von zwei schlechten Rauschgeneratoren, die voneinander abgeschirmt oder relativ weit entfernt sind). Das EXOR dieser Bit-Folgen ergibt eine Bit-Folge mit

.

Wahrscheinlichkeit	Ereignis	
$2/3 * 2/3 = 4/9$	$0 \; exor \; 0 = 0$	(3)
$2/3 * 1/3 = 2/9$	$0 \; exor \; 1 = 1$	(4)
$1/3 * 2/3 = 2/9$	$1 \; exor \; 0 = 1$	(5)
$1/3 * 1/3 = 1/9$	$1 \; exor \; 1 = 0$	(6)

.

Insgesamt also: Zu 5/9 Nullen und zu 4/9 Einsen; die Abweichung von der Gleichverteilung wurde durch das EXOR um 2/3 reduziert.

Modulo-2-Summe von n Zufallsbits und der zentrale Grenzwertsatz

Nach dem Zentralen Grenzwertsatz ist die Wahrscheinlichkeitsdichte der Summe s von $n \gg 1$ echt zufälligen Bits die Normalverteilung, weil eine Summe von Bits automatisch die Lindebergsche Bedingung erfüllt. Der zentrale Grenzwertsatz gilt zwar genau genommen nur für unendliches n, aber er ist meist schon bei ungefähr 10 in guter Näherung gültig.

Liegt das Maximum der Normalverteilung p_v bei exakt $m+0,5$ ($m \in \mathbb{N}$), dann ist die Amplitude der Normalverteilung bei $m+0,5-k/2$ dieselbe wie bei $m+0,5+k/2$. Weil die Modulo-2-Summe bei $m+0,5-k/2$ und $m+0,5+k/2$ einmal 1 und einmal 0 ist, bedeutet dies, dass sich für jeden k-Wert, und damit auch für die gesamte Summe, eine 50 % Wahrscheinlichkeit für eine 0 und eine 1 ergibt. Dies ist der "best case", denn es bedeutet, dass die Modulo-2-Summen-Bits (Sekundärbits) eine Entropie von 1,0 Bit/Bit besitzen.

Im "worst case" liegt das Maximum der Normalverteilung p_v bei m ($m \in \mathbb{N}$), und der Betrag der Differenz der Wahrscheinlichkeit, dass die Modulo-2-Summe null ist minus der Wahrscheinlichkeit, dass die Modulo-2-Summe eins ist, ergibt sich zu:

$$\mid p(0)-p(1) \mid_{worst\ case} = \frac{1}{\sigma \cdot \sqrt{2 \cdot \pi}} \cdot 2 \cdot \sum_{k=1}^{\infty} \left(e^{-\frac{(2 \cdot k-1)^2}{2 \cdot \sigma^2}} - e^{-\frac{(2 \cdot k)^2}{2 \cdot \sigma^2}} \right) \quad . \tag{7}$$

Die Standardabweichung σ ist proportional zu $\sqrt{n}$ ("ars conjectandi"), sodass $\sigma = c \cdot \sqrt{n}$ mit der Konstanten c eingesetzt werden kann:

$$\mid p(0)-p(1) \mid_{worst\ case} = \frac{\sqrt{2}}{c \cdot \sqrt{n \cdot \pi}} \cdot \sum_{k=1}^{\infty} \left(e^{-\frac{(2 \cdot k-1)^2}{2 \cdot c^2 \cdot n}} - e^{-\frac{(2 \cdot k)^2}{2 \cdot c^2 \cdot n}} \right) \quad . \tag{8}$$

Dieser Ausdruck sinkt streng monoton mit steigendem n und der Grenzwert für $n \to \infty$ ist null (Abb. 2).

Somit ergibt sich also auch im "worst case" aus mehreren primären echt zufälligen Bits mit wenig Entropie mittels Modulo-2-Addition ein sekundäres echt zufälliges Bit mit mehr Entropie.

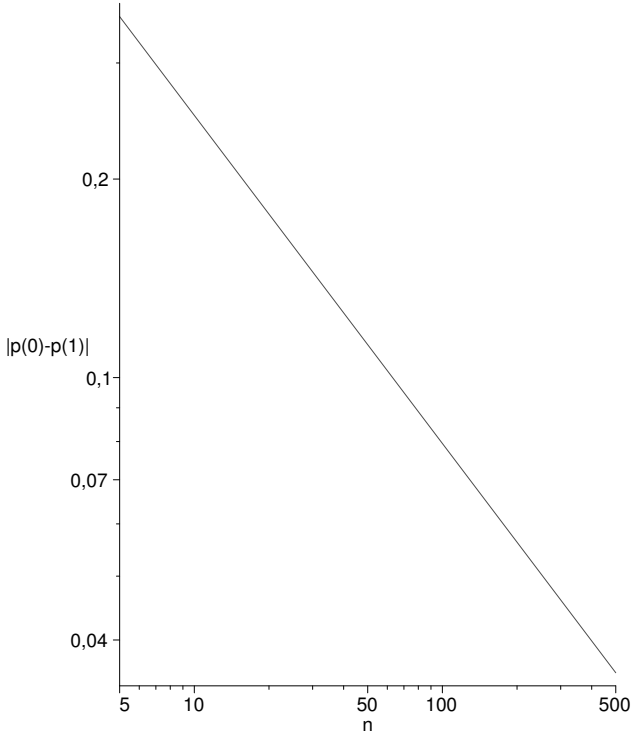


Abbildung 2: Entropiezuwachs durch Exklusiv-Oder-Verknüpfung von n Zufallsbits im “worst case”, aufgetragen über n.

Der Vorteil hierbei ist, dass die primären Bits nicht statistisch unabhängig sein müssen und zudem die Summe auch Pseudozufallsbits enthalten kann, wenn ausreichend viele echte Zufallsbits in der Summe enthalten sind. Dadurch können mit Pseudozufallsbitgeneratoren gezielt statistische Auffälligkeiten der echt zufälligen Bits beseitigt werden.
Weil die Gleichung 8 unabhängig von den einzelnen Bits gilt, werden durch die Modulo-2-Addition auch die

Korrelationen innerhalb der sekundären Bitsequenz abgeschwächt, sodass auch die Markov-Entropien erhöht werden.

3 Praxis

Im Prinzip kann eine azyklische Zufallsbitfolge schon dadurch erzeugt werden, dass ein Signal mit der (Wiederhol-)Frequenz f_1 von einem Oszillator erzeugt wird und dieses Signal mit einer Frequenz f_2 von einem anderen Oszillator abgetastet wird. Diese Abtastwerte in digitaler serieller Form bilden eine Zufallsbitfolge, die azyklisch ist, wenn f_1 und f_2 inkommensurabel sind, es also keine natürlichen Zahlen n und m gibt, sodass $n \cdot f_1 = m \cdot f_2$ erfüllbar ist. Diese Bedingung ist gleichbedeutend damit, dass $\frac{f_1}{f_2}$ keine rationale Zahl ist und deshalb fast immer erfüllt ist, weil f_1 und f_2 im Allgemeinen irgendwelche reellen Zahlen sind (woraus folgt, dass auch $\frac{f_1}{f_2}$ irgendeine reelle Zahl ist), und weil bekanntlich fast alle reellen Zahlen irrational sind, da die Menge der rationalen Zahlen eine Lebesgue-Nullmenge ist.
Obwohl also die Wahrscheinlichkeit für $n \cdot f_1 = m \cdot f_2$ gleich null ist, zeigt sich in der Praxis aber, dass die Qualität der so erzeugten Zufallszahlen relativ schlecht ist, weil sie im Wesentlichen von dem Rauschverhalten der beiden Oszillatoren bestimmt wird, und Oszillatoren in der Regel relativ rauscharm sind.
Deshalb sind, in einer Taktperiode, meist viele solche Zufallsbits nötig um mittels EXOR, das normalerweise mit einem Paritätsgenerator berechnet wird, ein (nahezu) perfekt zufälliges sekundäres Zufallsbit zu erzeugen. Um den Aufwand gering zu halten, können einige wenige der primären Zufallsbits auch Pseudozufalls-Bits aus Schieberegister-Generatoren sein, weil sie die Qualität der sekundären Zufallsbits nicht verschlechtern (siehe Lindeberg-Kriterium).
Eine andere Möglichkeit ist Rekursion, die in vielen Variationen mit wenig Aufwand realisiert werden kann. Im einfachsten Fall wird einfach das sekundäre Zufallsbit über ein D-Flip-Flop auf den Parätsgenerator gegeben, der das sekundäre Zufallsbit berechnet.

Löten

Der Aufbau von solchen echten Zufallszahlengeneratoren ist sehr einfach, weil sie 100 % digital sind. Als digitale Rauschquelle dient z. B. ein invertierender Schmitt-Trigger wie 1/6 74HCT14, dem ein D-Flip-Flopp wie z. B. 1/8 74ACT157 nachgeschaltet ist, um die Zufallsbits synchron zum Takt weiter verarbeiten zu können. Beim invertierenden Schmitt-Trigger (IST) ist der Eingang auf den Ausgang zu löten (Ringoszillator der Länge 1) und brachliegende ISTs können als Ausgangsstufe mit zusätzlicher Phasenverschiebung genommen werden. Von einem IC wie dem 74HCT14 könen zumindest zwei Schmitt-Trigger verwendet werden, weil die Schmitt-Trigger zumindest in solchen ICs halbwegs unabhängig voneinander mit ca. 90 MHz schwingen. Die so erzeugten primären Zufallsbits gehen anschließend auf den Paritätsgenerator, z. B. 1/9 74F280 und hiernach wieder auf ein D-Flip-Flop wie z. B. 1/8 74ACT157. Eine kleine Version mit bis zu 6 primären Zufallsbits bekommt man also schon mit einem 74HCT14, 74ACT157 und einem 74F280 für rund ein Euro hin und kann damit sogar noch Varianten wie Rückkoppeln des Ausgangs auf den Paritätsgenerator mit einem Takt Verzögerung realisieren.

4 Schluß

Zum Testen von echten Zufallszahlengeneratoren eignen sich nicht nur Tests mit Software sondern auch Auto- und Kreuzkorrelatoren sowie FFT-Spektrometer, die im selben Chip integriert werden können um damit beispielsweise eine Echtzeit-Selbstdiagnose vorzunehmen, die auch Chip-intern geloggt werden kann.

Die zuverlässige und schnelle Erzeugung von echten Zufallszahlen ist deshalb sowohl theoretisch als auch technisch weder schwer noch teuer, aber trotzdem werden in der Praxis echte Zufallszahlengeneratoren sehr selten eingesetzt, weil entweder Pseudozufallszahlengeneratoren meist ausreichen oder /dev/random schnell genug ist.
Deshalb ist das Herstellen von Zufallszahlengeneratoren eine brotlose Kunst, denn es gibt dafür relativ wenig Nachfrage, die zu 50 % von Parapsychologen mit schlechter Zahlungsmoral stammt.
Die Firma, in der ich nach der Uni gearbeitet habe und die auch die Zufallszahlengeneratoren verwendet hat, Kryptografix, ging schon nach 6 Monaten in Konkurs und derzeit suche ich auch wieder einen Job!

5 Literatur, Links

- Mail-Adr. auch für Kommentare u. Anregungen: rolf.freitag@email.de
- Literatur: Buch Linux Device Drivers / Linux Gerätetreiber, Buch Anwendungen entwickeln unter Linux, Buch Linux Treiber Entwickeln, (Kernel-)Sourcen, Howtos wie das Linux I/O port programming mini-HOWTO.
- Link z. Vortrag: ftp://random.linux-site.net/21C3/
- Dissertation mit Kapiteln zum Thema: http://www.nefkom.net/nobody/doct.pdf
  Mirror: ftp://random.linux-site.net/doct/doct.pdf
- Kurze Beschreibung einer meßtechnischen Anwendung von Zufallszahlengeneratoren:
  http://web.archive.org/web/20010404075108/hlhp1.physik.uni-ulm.de/~freitag/
- Einige meiner Zufallszahlengeneratoren:
  http://web.archive.org/web/20010415015513/hlhp1.physik.uni-ulm.de/~freitag/Spinoffs.html
- Kleiner Artikel zur Theorie in Englisch:
  ftp://random.linux-site.net/random/exor.pdf
  ftp://random.linux-site.net/doct/azycl1.pdf
- Patent auch mit Schaltplänen:
  ftp://random.linux-site.net/pat/zufpat1.jpg
  ftp://random.linux-site.net/pat/zufpat2.jpg
  ftp://random.linux-site.net/pat/zufpat3.jpg
  ftp://random.linux-site.net/pat/zufpat4.jpg
  ftp://random.linux-site.net/pat/zufpat5.jpg
- Diehard-Tests für Zufallszahlen: http://stat.fsu.edu/~geo/diehard.html

- Fips-Tests für Zufallszahlen: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

Anmerkung: random.linux-site.net ist langsam weil stark limitiert bezüglich Gesam-Bandbreite, Bandbreite pro Verbinung, Anzahl Gesamt-Verbindung und Anzahl der Verbingungen pro User, so dass ein Download-Manager wie wget empfohlen wird.

6 Experimentalzirkus

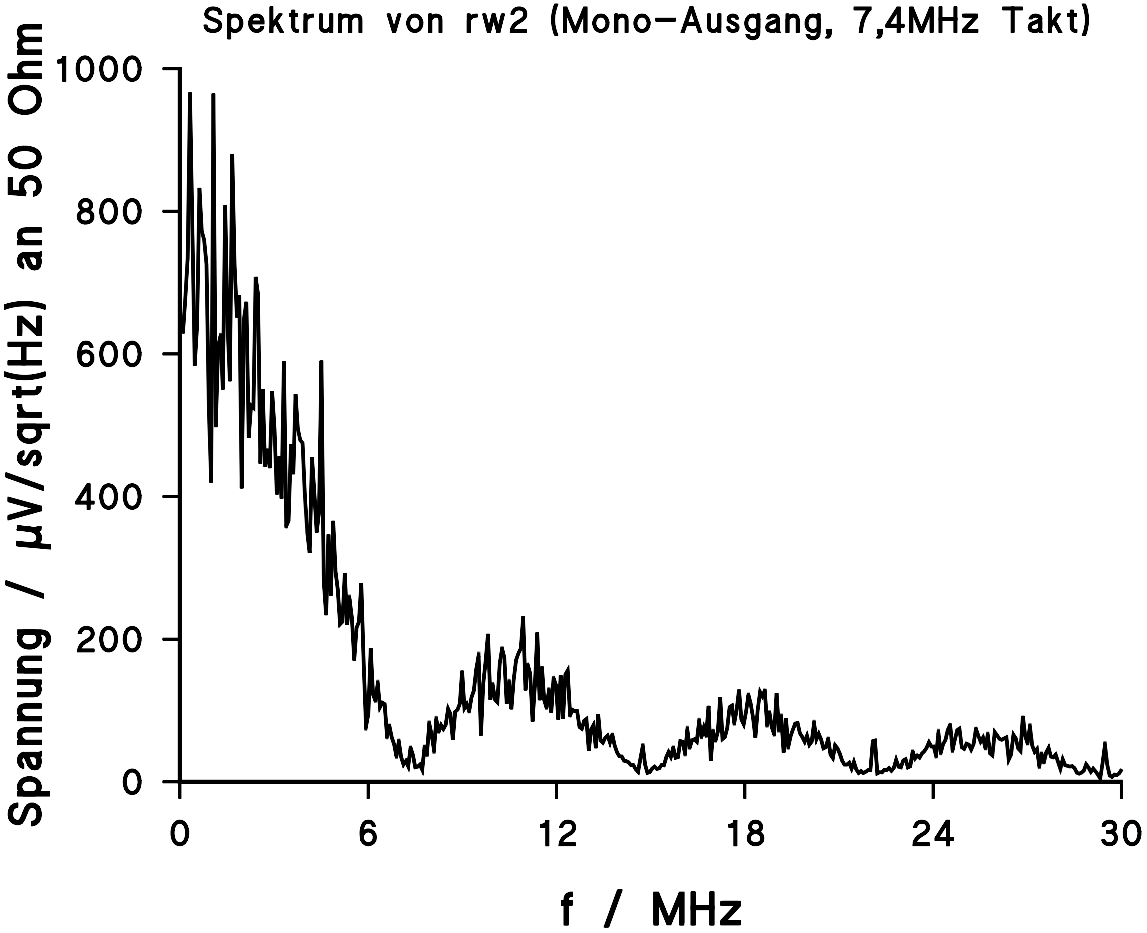
7 Anhang

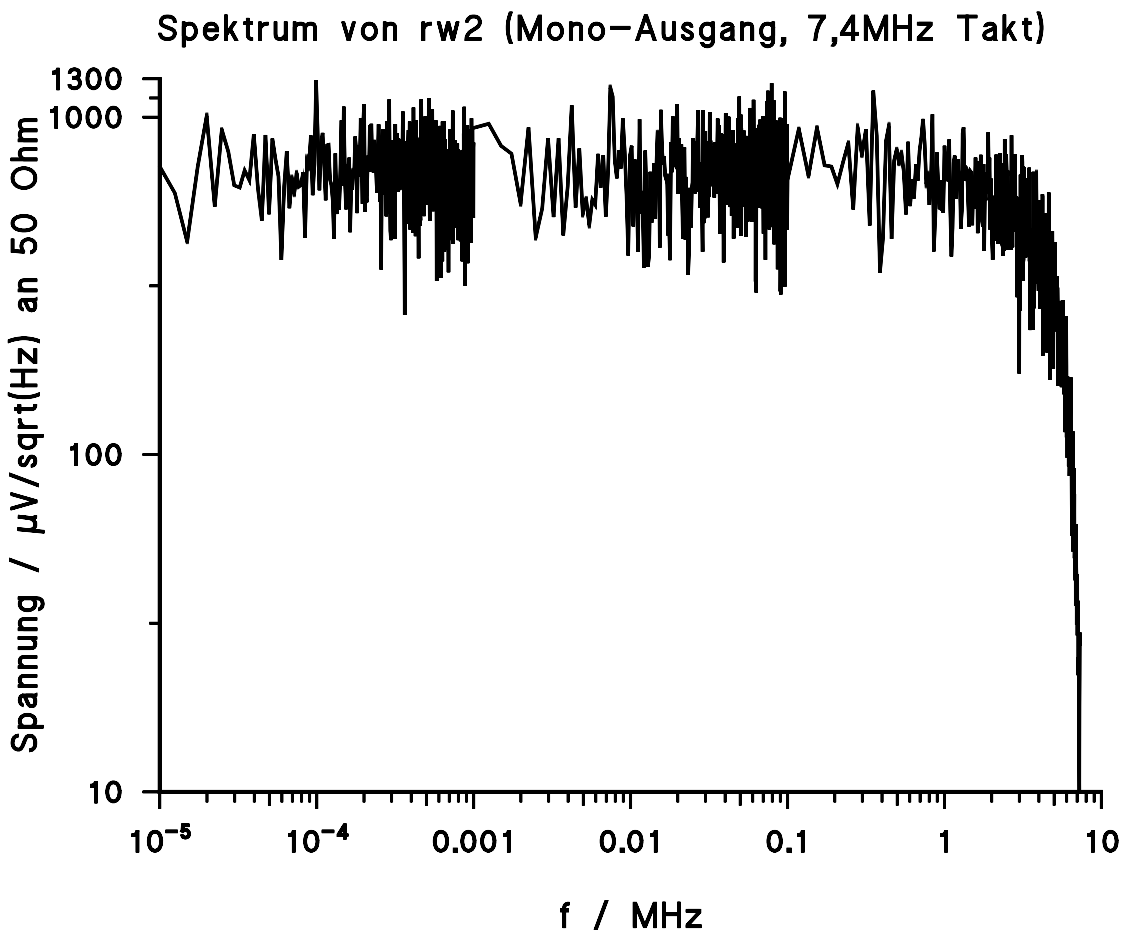
Entropiebelag: Die Entropie pro Symbol in einer Markow-Kette, denn ein echter Zufallszahlengenerator ist ein Markow-Prozess, siehe H. Völz: *Grundlagen der Information*, Akademie Verlag, Berlin, 1991, Kapitel 1.2.5.. Praktisch verwendet werden aber nur Tests, die einige Eigenschaften relativ kurzer Zufallszahlenreihen testen und wegen den statistischen Fluktuationen zwangsläufig immer mal wieder falsch-positive und falsch-negative Testresultate liefern.

Kontinuierliches Amplituden-Spektrum eines perfekten Zufallsbitgenerators

$$S(f) = \sum_{n=1}^{\infty} \frac{2^{(-n)} \left| \frac{sin(\frac{n \cdot \pi \cdot f}{f_0})}{f} \right|}{\pi} \tag{9}$$

Es ist im Wesentlichen ein sin(f/f_0)/f-Sprektrum mit f_0=Taktfrequenz.


Spektrum von rw2 (Mono-Ausgang, 7,4MHz Takt)
Spannung / µV/sqrt(Hz) an 50 Ohm
1000
800
600
400
200
0
0
6
12
18
24
30
f / MHz


Spektrum von rw2 (Mono−Ausgang, 7,4MHz Takt)
Spannung / µV/sqrt(Hz) an 50 Ohm
1300
1000
100
10
10^-5
10^-4
0.001
0.01
0.1
1
10
f / MHz