DESIGN AND PERFORMANCE ANALYSIS OF A HIGH SPEED AWGN COMMUNICATION CHANNEL EMULATOR

L.E.S.T.E.R
Laboratoire d'Electronique des Systèmes TEmps Réel

Prof. Emmanuel Boutillon
LESTER
South Brittany University

UBS
UNIVERSITÉ DE BRETAGNE-SUD

emmanuel.boutillon@univ-ubs.fr

Collaboration

1995: ENST Paris,
UofT (Glenn Gulak)

2000-2001:
ENST Paris (France),
SUP'COM Tunis (Tunisia),
LESTER, Lorient (France)

Outline

I Introduction

II Previous White Gaussian Noise Generator

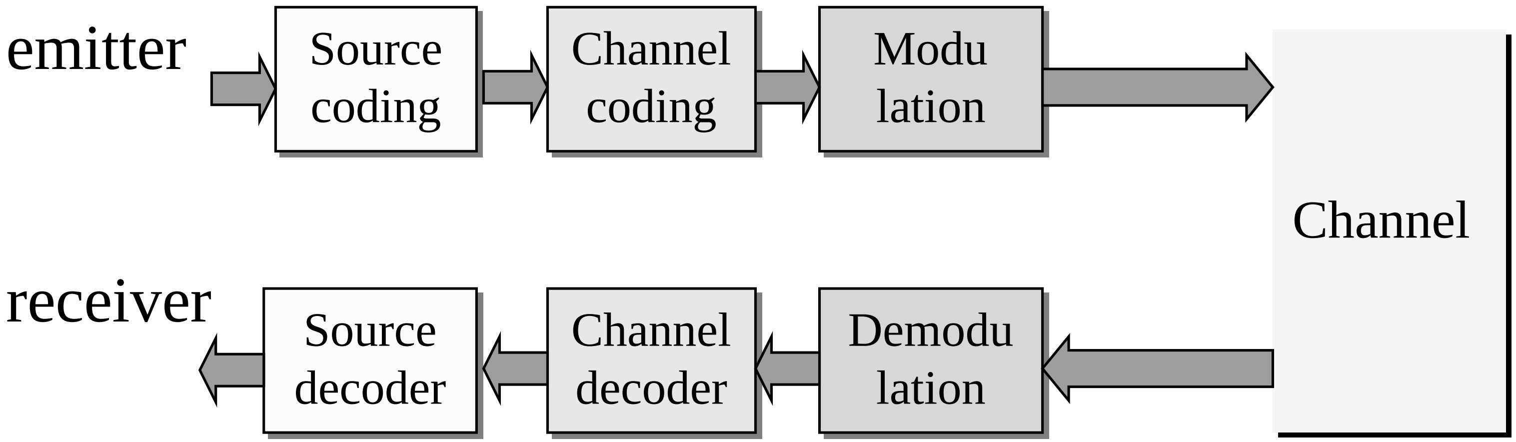
III Proposed WGNG

IV Hardware architecture design

V Conclusion

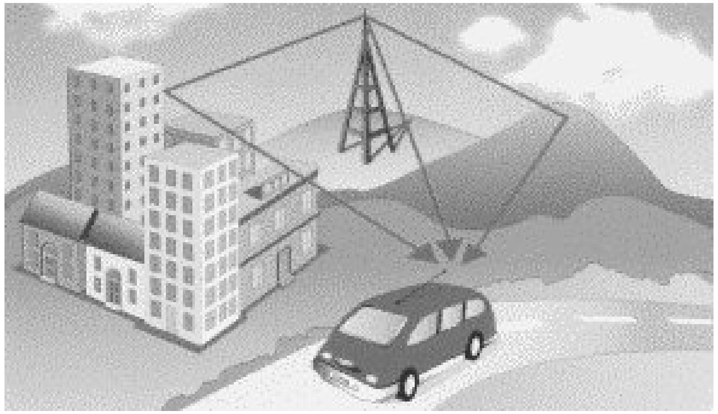
Motivations

Design of a communication system...



...find the best complexity-performance trade-off

Motivation



Performance:

- BER
- Jammer rejection
- time of synchronization…
- …

Complexity:

- area, power dissipation
- time to market
- …

algorithm
ADC resolution,
sampling frequency,
fixed precision

A very complex problem...

Monte-carlo simulation

* Formal expression of the BER: refer to Proakis

* In practice, estimation of the BER using Monte-carlo simulation

1) Software model of emitter, channel, receiver
2) Emulation of the transmission of N bits
3) Estimation of the BER as Nb_errors/N

VERY FLEXIBLE

but...

TIME CONSUMMING: BER of 10^{-6} (+-3%) requires 10^{9} bits.

Software simulation

Three methods to reduce the simulation time:

a) code optimization

b) powerful computing

c) parallel computing
(One Mbps for a turbo-decoder with a cluster of 16 PCs)

also use hardware emulation

Current methodology

Software	*Hardware*
Algorithm	VHDL programs
C programs	Synthesis, place and route operations
Compilation	Validation
Validation/optimization with long simulations	Final prototype
Fix specifications	

Proposed methodology

Software

Algorithm

C programs

Compilation

Validation/optimization

Fix algorithm + Set of non-specified parameters

Hardware

Generic VHDL programs, IP

Synthesis, place and route operations (on FPGA)

Hardware simulation/validation

Final prototype

Channel emulation

Type of communication channel:

- AWGN
- Rice
- Rayleigh

...

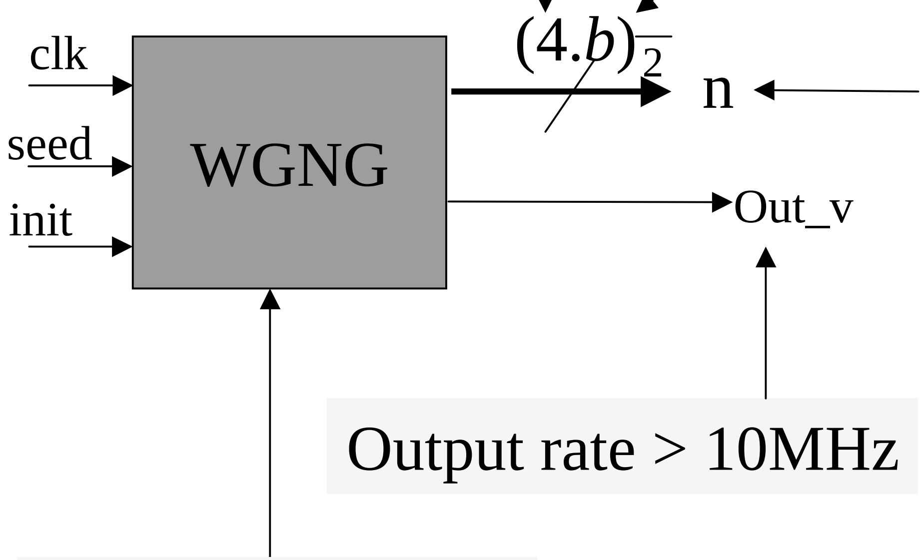
All those channels can be derived from Gaussian Noise (with ARMA filter, non-linear operators).

=> Need a White Gaussian Noise Generator (WGNG)

Specifications of the WGNG

b=2 to 10 bit after the dot

Sample between -8 and 8



a Flat spectrum

a (4σ, 1%) normal-like p.d.f

$$\xi_X(x) = \frac{X(x) - N(0,1)(x)}{N(0,1)(x)}$$

$|\xi_X(x)| < 1\%$ for $|x| < 4$

Output rate > 10MHz

A periodicity > 2^{60}

+ LOW COST FPGA IMPLEMENTATION

Outline

I Introduction

II Previous White Gaussian Noise Generator

III Proposed WGNG

IV Hardware architecture design

V Conclusion

Previous WGNG

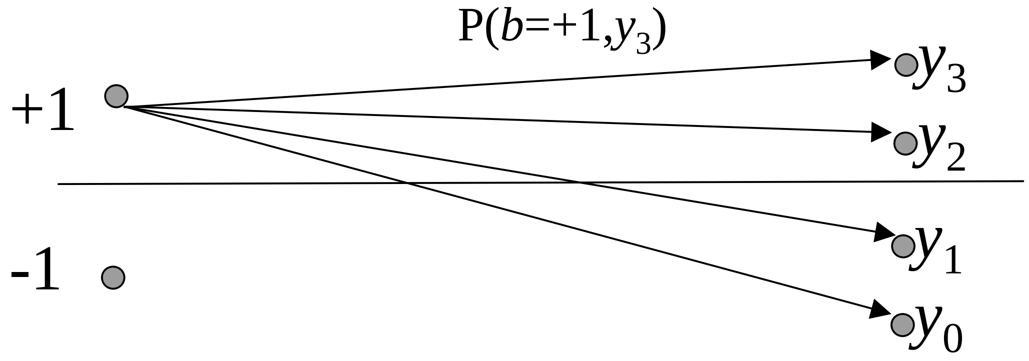
0) Using thermal noise of a resistor (non deterministic)

1) Case of low ADC precision

2) Central limit theorem

3) Box-Muller method

Case of low ADC precision (1)



Emitter

N=4-Level ADC of the receiver

The probabilities $P(x=i, y_j)$ are known for a given SNR

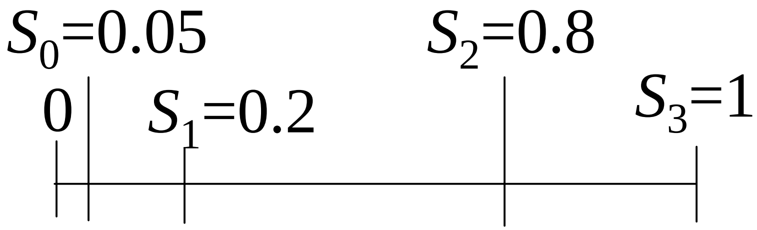
Example: $P(b=+1, y_3) = 0,3$ $P(b=+1, y_2) = 0,5$

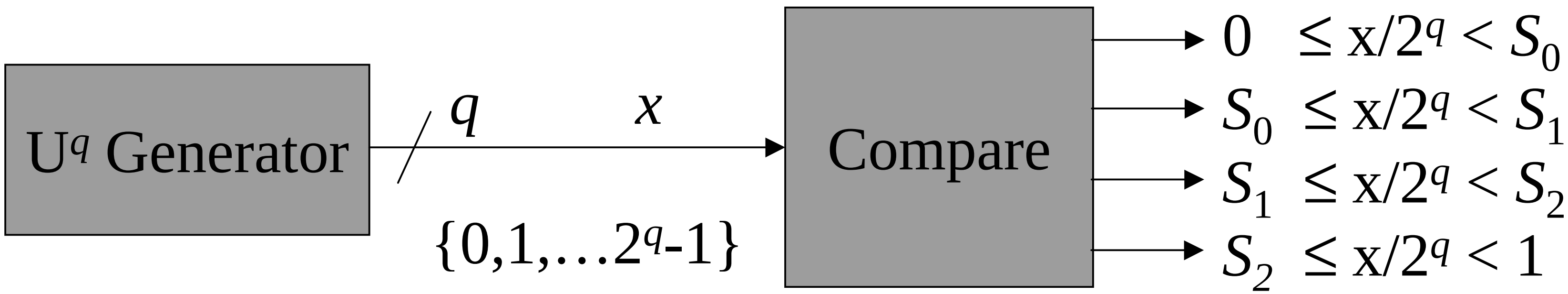
$P(b=+1, y_1) = 0,15$ $P(b=+1, y_3) = 0,05$

Case of low ADC precision (2)

Repartition function: $S_k = \sum_{j=0}^{k} P(x = i, y = j)$

Segment [0,1]





Precision depends on q, Complexity in $O(qN)$

Central limit theorem

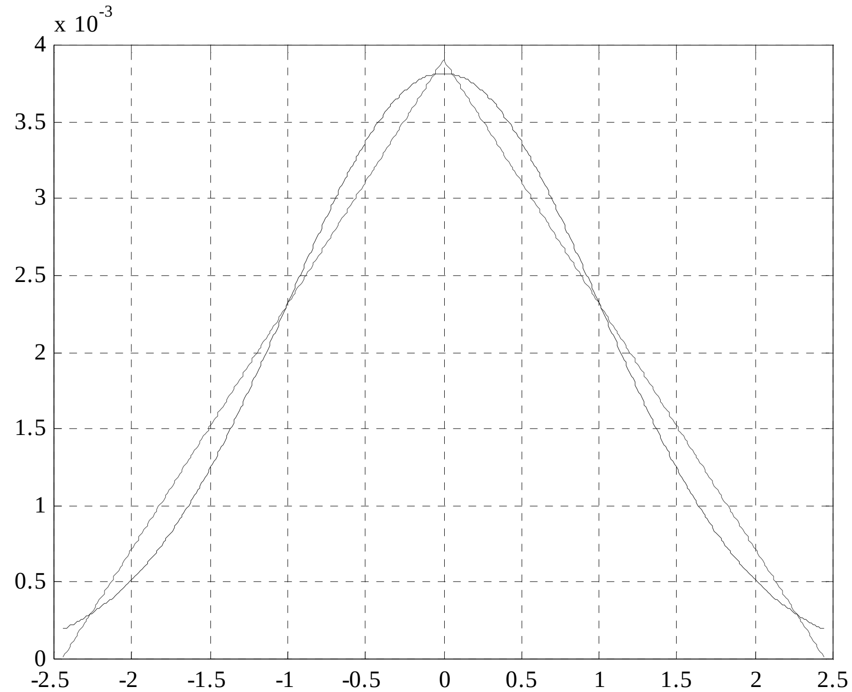
X is a real r.v. of mean m_x and standard deviation σ_x,

X_N defined as: $$X_N = \frac{1}{\sigma_x \sqrt{N}} \sum_{i=0}^{N-1} (x_i - m_x)$$

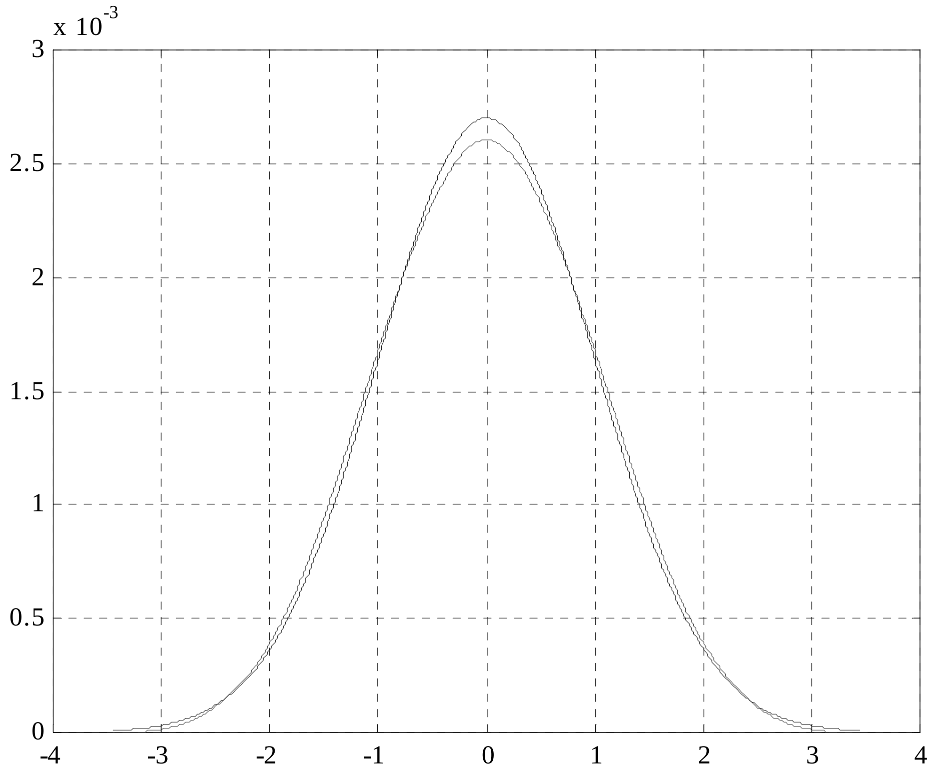
tends towards $N(0,1)$, when N tends towards infinity.

Let U(q,N) = sum of N U^q, (Uniform distribution over $\{0,\ldots,2^q-1\}$)

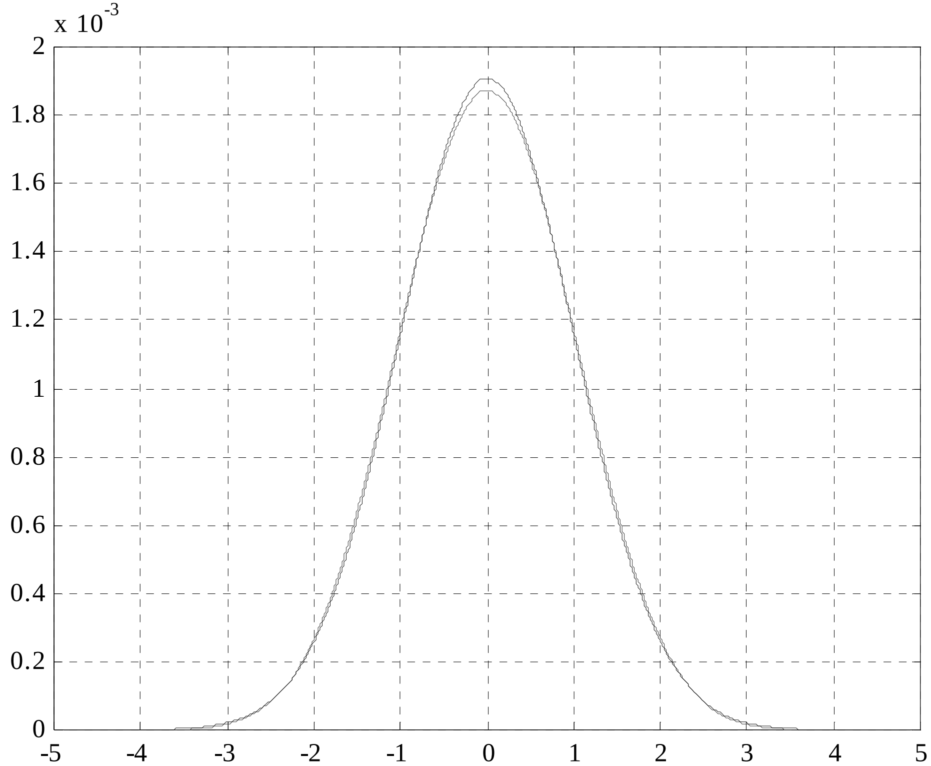
P.d.f. U(q=8,N=2)



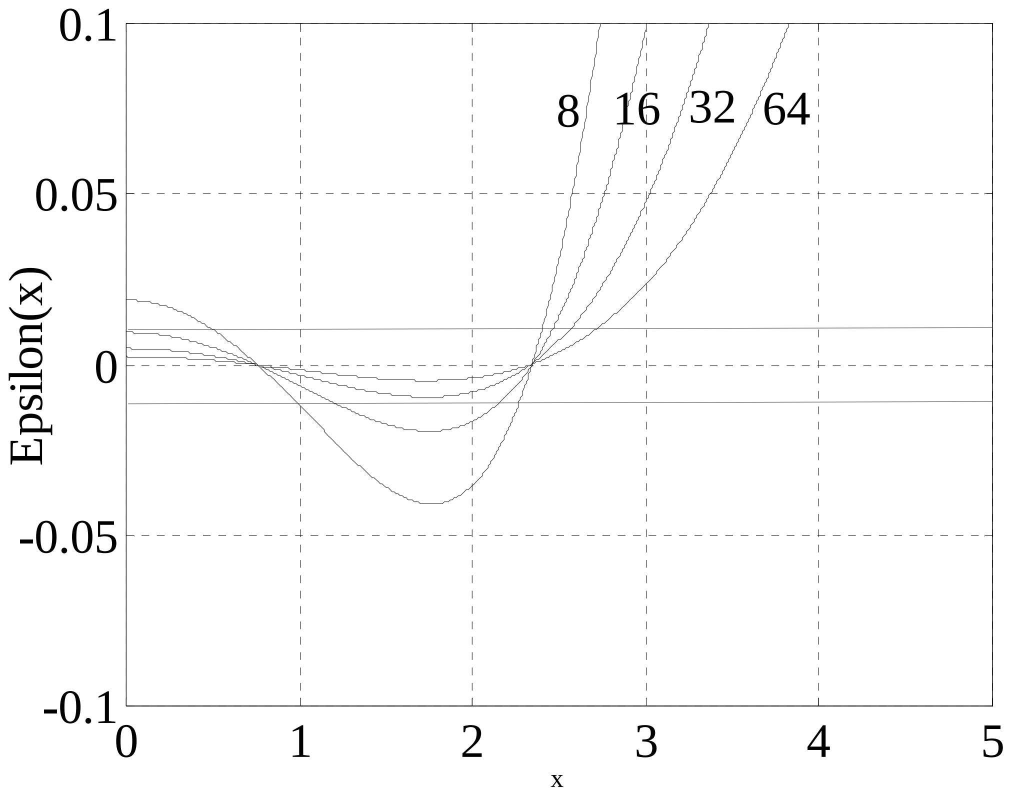
P.d.f. of U(q=8,N=4)



P.d.f U(q=8,N=8)



Epsilon function



The convergence is very slow...

Box-Muller method

Method used in software program:

If x_1 and x_2 are two uniform r.v. over [0,1], then:

$$f(x_1) = \sqrt{-\ln(x_1)}$$
$$g(x_2) = \sqrt{2}\cos(2\pi x_2)$$
$$n = f(x_1)g(x_2)$$

give a sample n of the normal distribution

Efficient with a floating CPU unit, not with an FPGA

Outline

I Introduction

II Previous White Gaussian Noise Generator

III Proposed WGNG

IV Hardware architecture design

V Conclusion

Proposed method

Quantized version of Box-Muller method adapted to hardware implementation

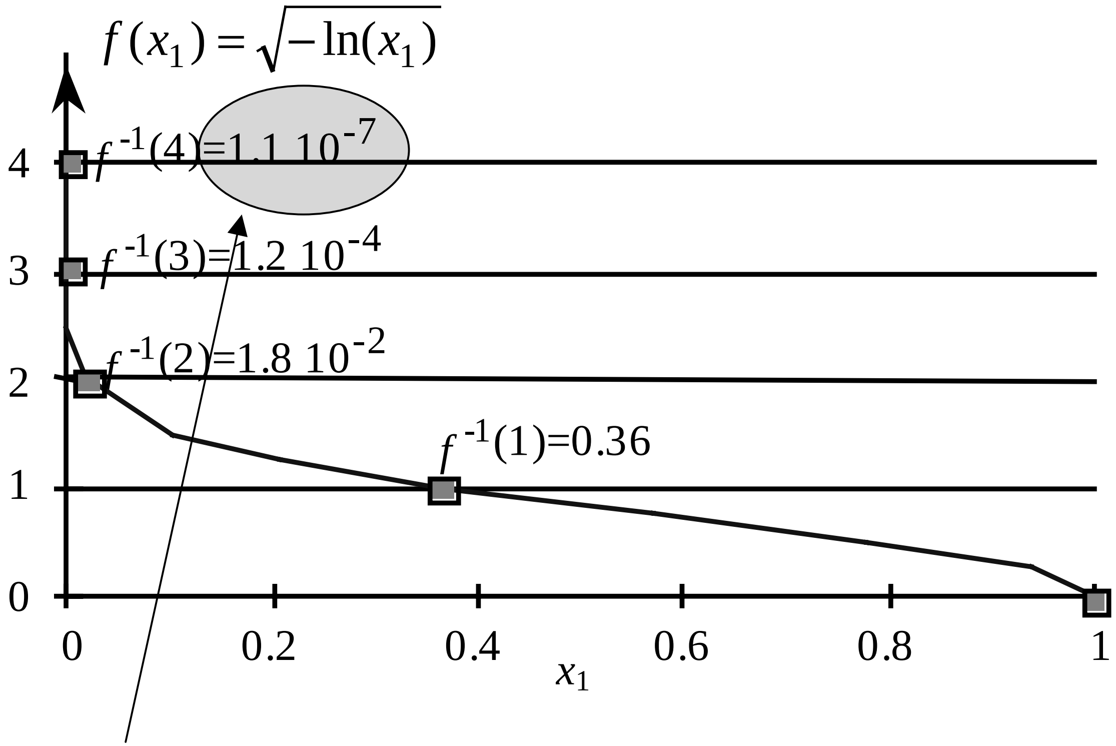
=> rough distribution

Smooth the distribution using central limit theorem

Desire an accurate complexity model and an exact distribution

Quantization of $f(x_1)$

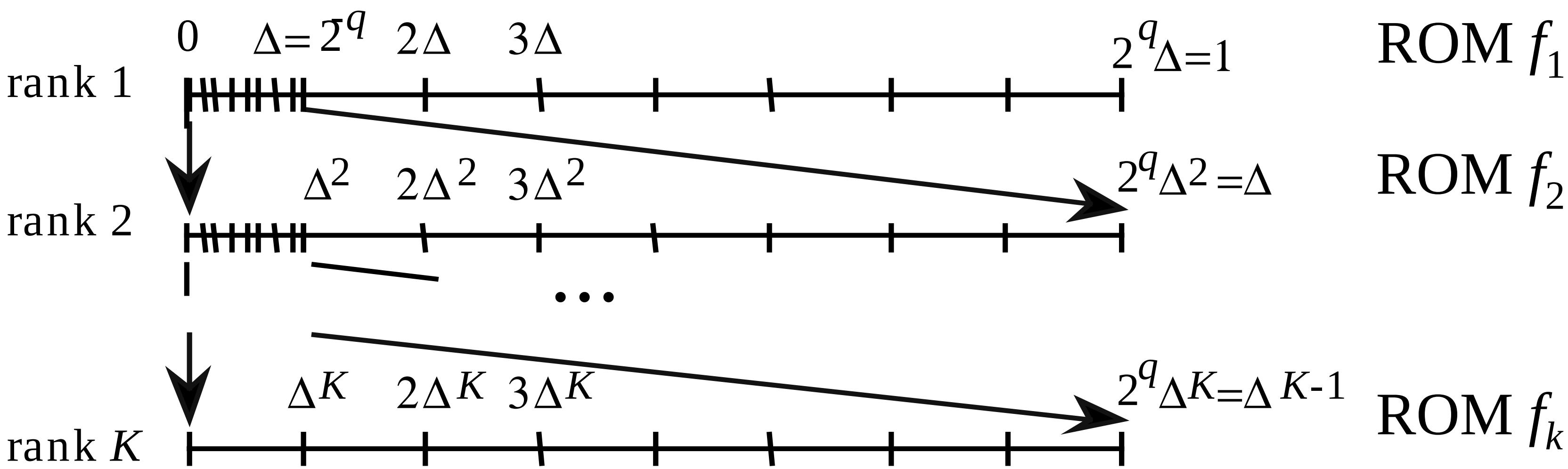
Plot of function $f(x_1)$



Need a fine quantization around 0

Non uniform quantization (1)

Let s_1, s_2, …, s_K be K independent r.v. of q bits (distribution U^q)



If s_1>0, use ROM f_1, else if s_2>0, use ROM f_2 … and so on...

Result: the probability to draw segment s of rank r is 2^{-rq}

Pre-compute values of the ROMs

The quantized value associated with the ROM r at the address s is:

$$f(x_1) = \sqrt{-\ln(x_1)}$$

$$f_r(s) = \left\lfloor 2^m \sqrt{\ln\left((s+\delta)\Delta^r\right)} \right\rfloor \quad (\times 2^{-m})$$

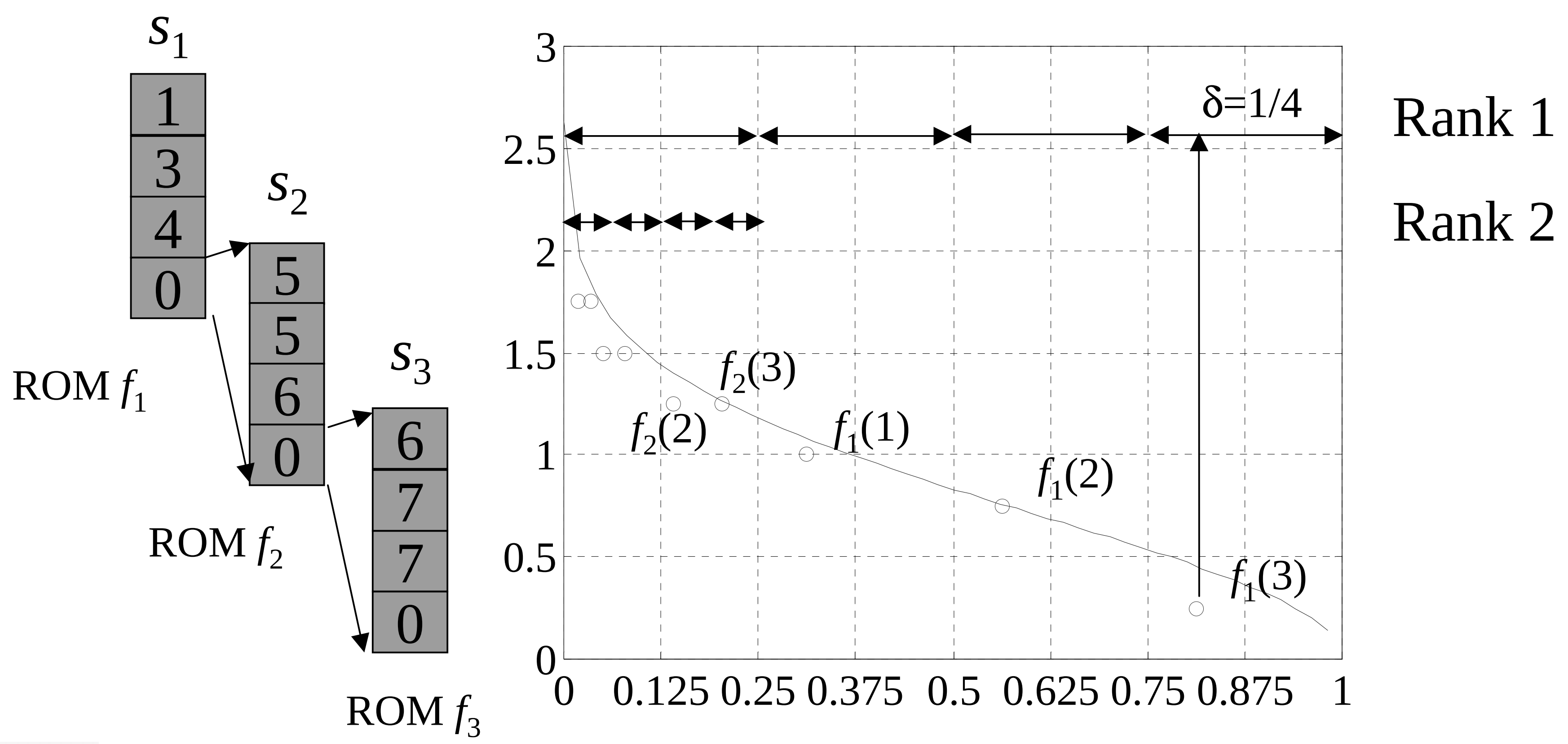
3 bits • m bits

δ relative position of x_1 in segment $[s\Delta^r, (s+1)\Delta^r[$

Remark: Probability to draw $f_r(s)$ is $P(f_r(s)) = 2^{-rq}$

Example of quantization of $f(x_1)$

K=3, q=2, m=2, δ=1/4



Quantization of $g(x_2)$

Let us define s', a q' bit random variable

$\Delta' = 2^{-q'}$ is the quantization step of segment [0,1/4]

ROM $g(s')$ is quantized as:

$$g(s') = \left\lfloor 2^{m'} \sqrt{2} \cos\left(\frac{\pi \Delta' (s' + \delta')}{2} \right) \right\rfloor \qquad (\times 2^{-m'})$$

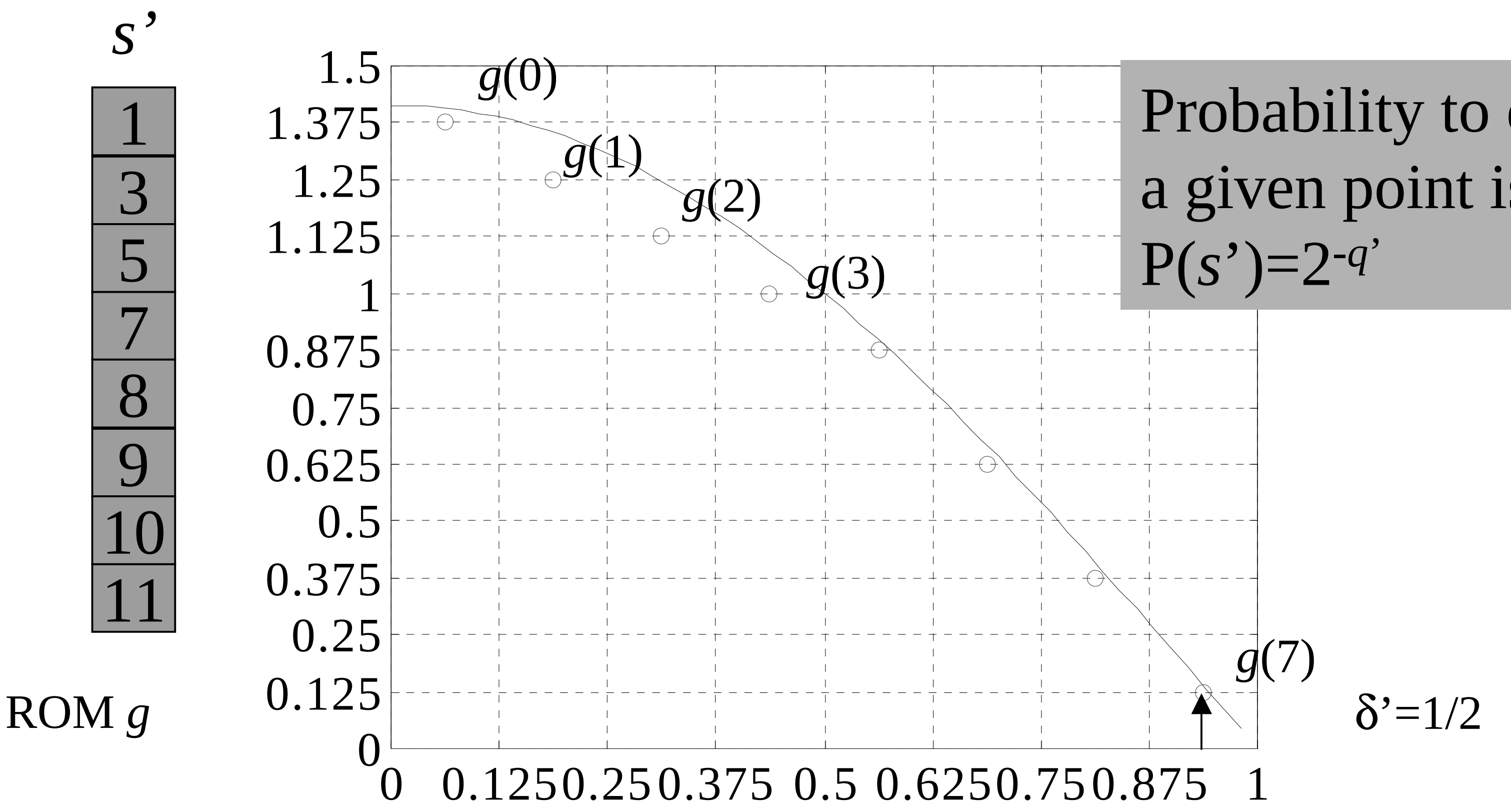
1 bit .	m' bits

δ' relative position of the point in segment $[s'\Delta', (s'+1)\Delta'[$

The problem of sign is analyzed later

Example of quantization of $g(x_1)$

$q'=3$, $m'=3$, $\delta'=1/2$



Probability to draw a given point is $P(s')=2^{-q'}$

Half Box-Muller r.v.

For a given triple (s,r,s'), n^+ (Half Box Muller) is computed as:

$$n^+ = \left\lfloor \frac{f_r(s) \times g(s')}{2^{m+m-b}} \right\rfloor \quad (\times 2^{-b}) \qquad (*)$$

Let S_n be the subset of $\{0, ..., 2^q\text{-}1\}\text{x}\{1, ..., K\}\text{x}\{0,..., 2^{q'}\text{-}1\}$ of all triples (s,r,s') that give n^+ using (*)

$$P(f_r(s), g(s')) = 2^{-(rq+q')}$$

$$P(HBM = n^+) = \sum_{(s,r,s')\in S_n} P(fr(s), g(s'))$$

The exact probability density function of *HBM* can be computed

Construction of HBM

```
scaling = pow2(m_f + m_g - b);

for s=1:pow2(q_f)-1
  for r=1:K
    for u=1:pow2(q_g)
   n=floor((rom_f(s,r)*rom_g(u)/scaling);
   HBM(n+1) = HBM(n+1) + pow2(-(r*q_f + q_g));
   end;
 end;
end;
```

Exhaustive exploration

Probability of the triplet s,r,u

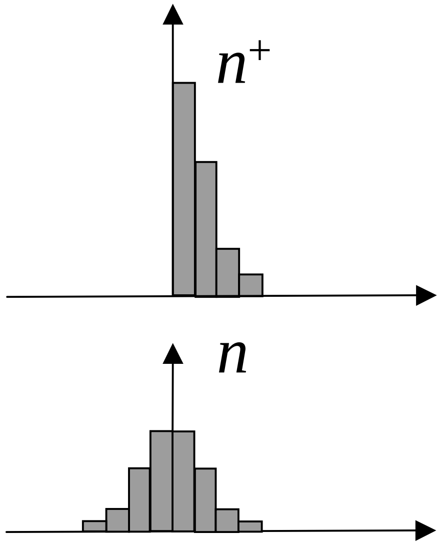
Box_Muller r.v.

From a binary r.v. *sign*, Box-Muller p.d.f. is obtained

HBM

$$n = (1-2\,sign)\; n^{+} - sign$$

BM



The exact p.d.f. of *BM* can also be computed

Example of distribution

Parameters:

b=6 bits after dot

K=5 f_r ROMs

q=4 (16 words ROM for f_r)

q'=8 (256 words ROM for g)

m=7 (3+m=10 bit-word for f_r)

m'=6 (1+m'=7 bit-word for g)

δ=0.36 δ'=0.5

Complexity:

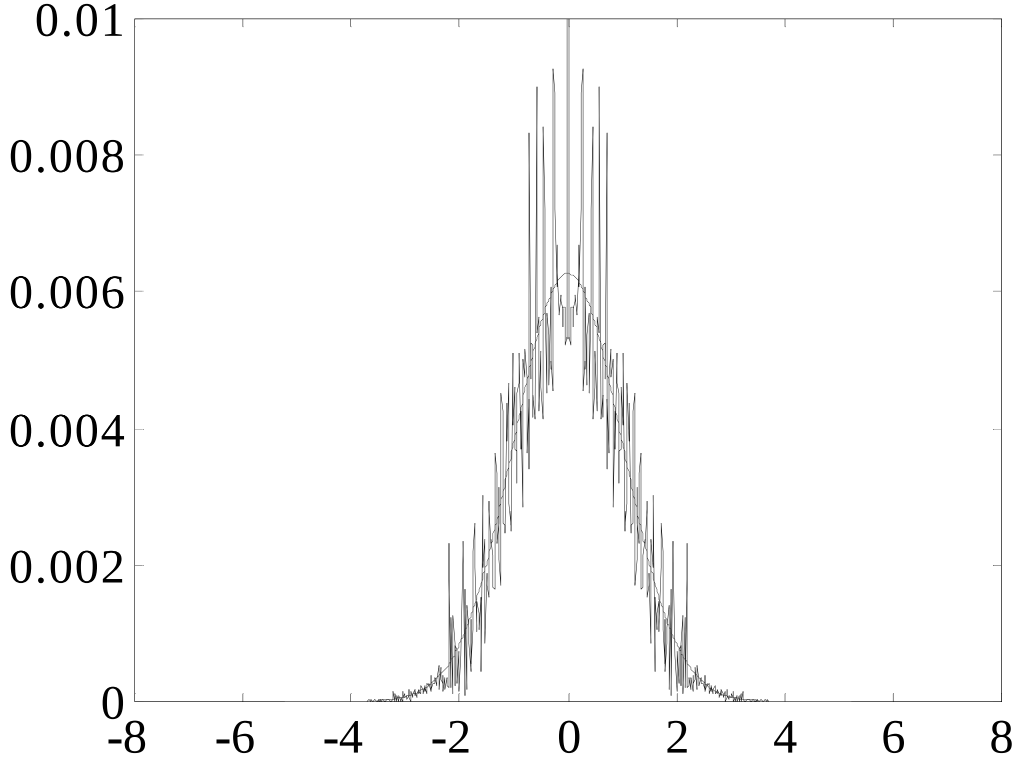
5 ROMs 16x10 for f_r

1 ROMs 256x7 for g

5x4 + 8 +1 = 29 binary r.v.

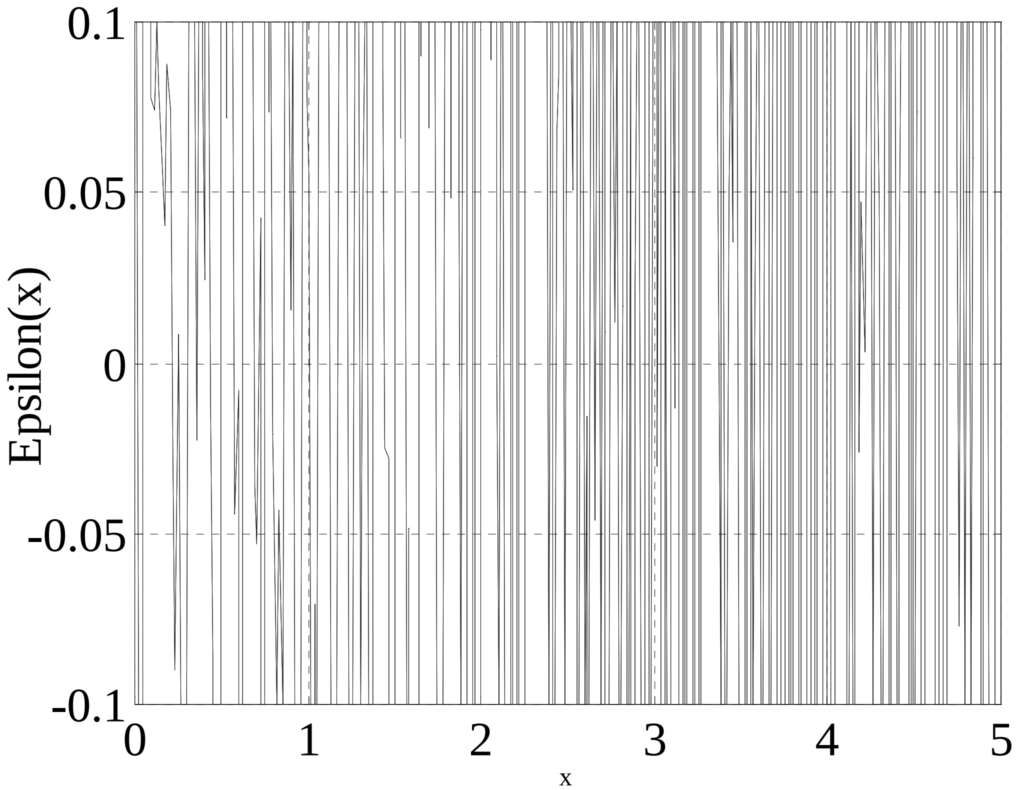
10 bits x 7 bits multiplier

Resulting p.d.f.



Large variations around N(0,1) due to quantization effects

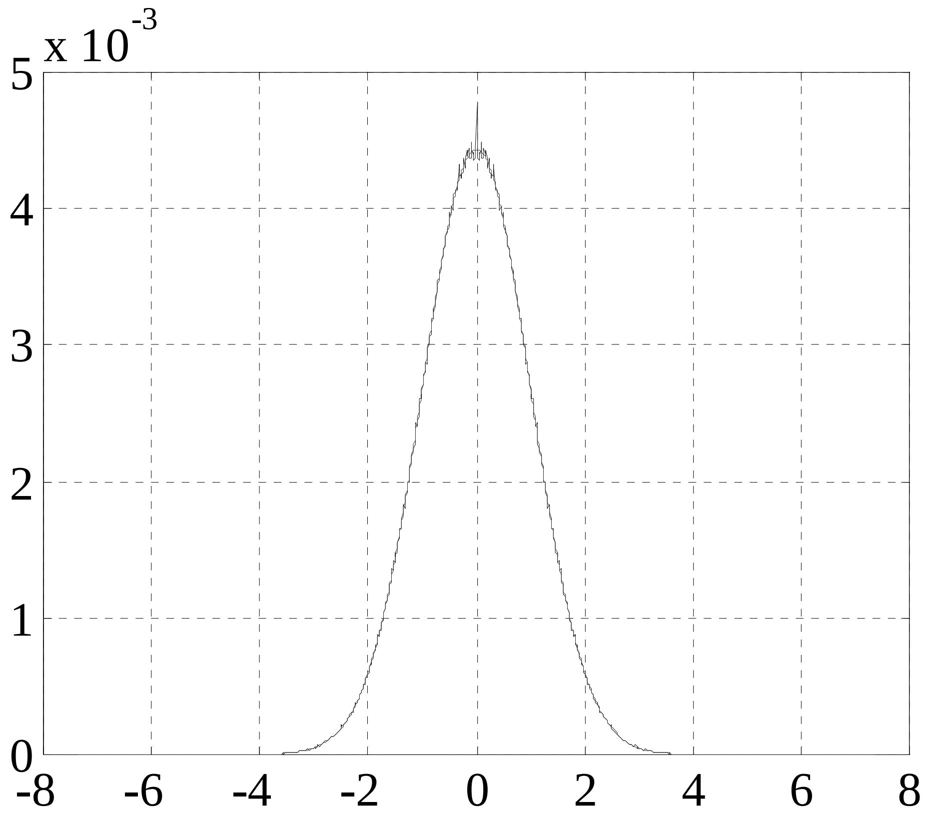
Epsilon function



Need to smooth the variation with central limit theorem

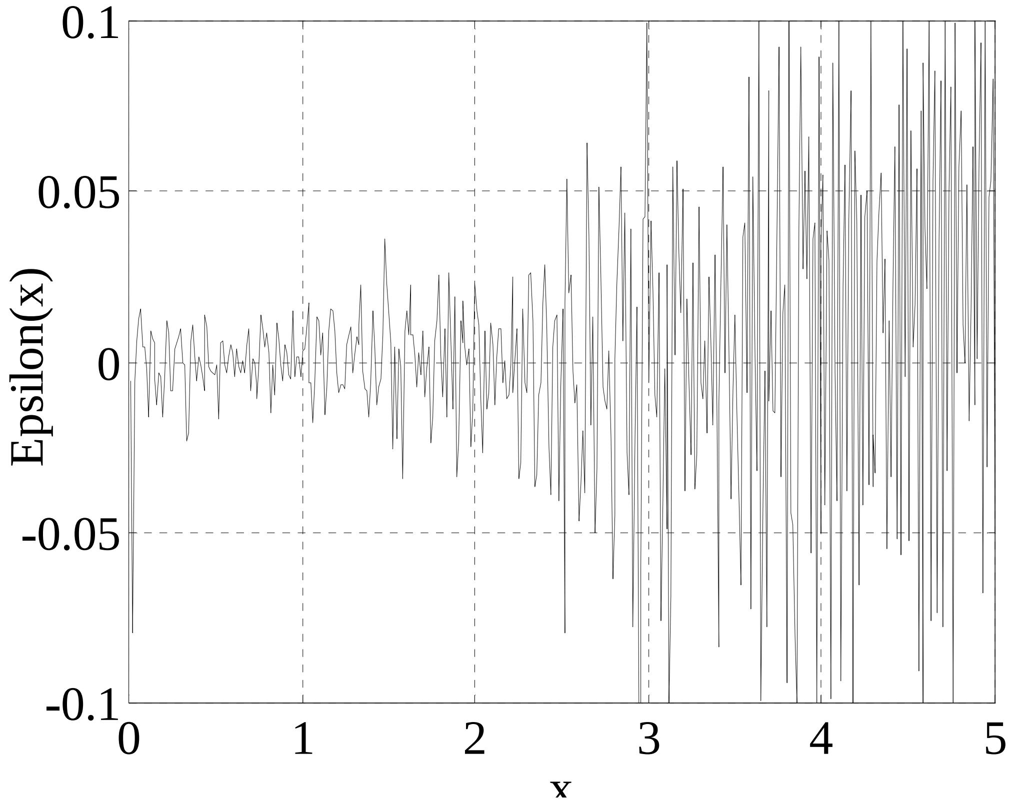
Use of central limit theorem

Generation of BM_2 as the sum of two independent draws of BM_1



Distribution of BM_2 can be computed ($BM_2 = BM_1 \otimes BM_1$)

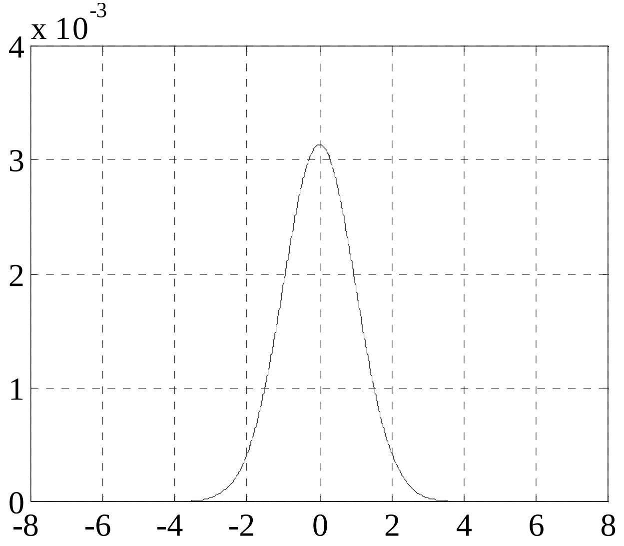
Use of central limit theorem

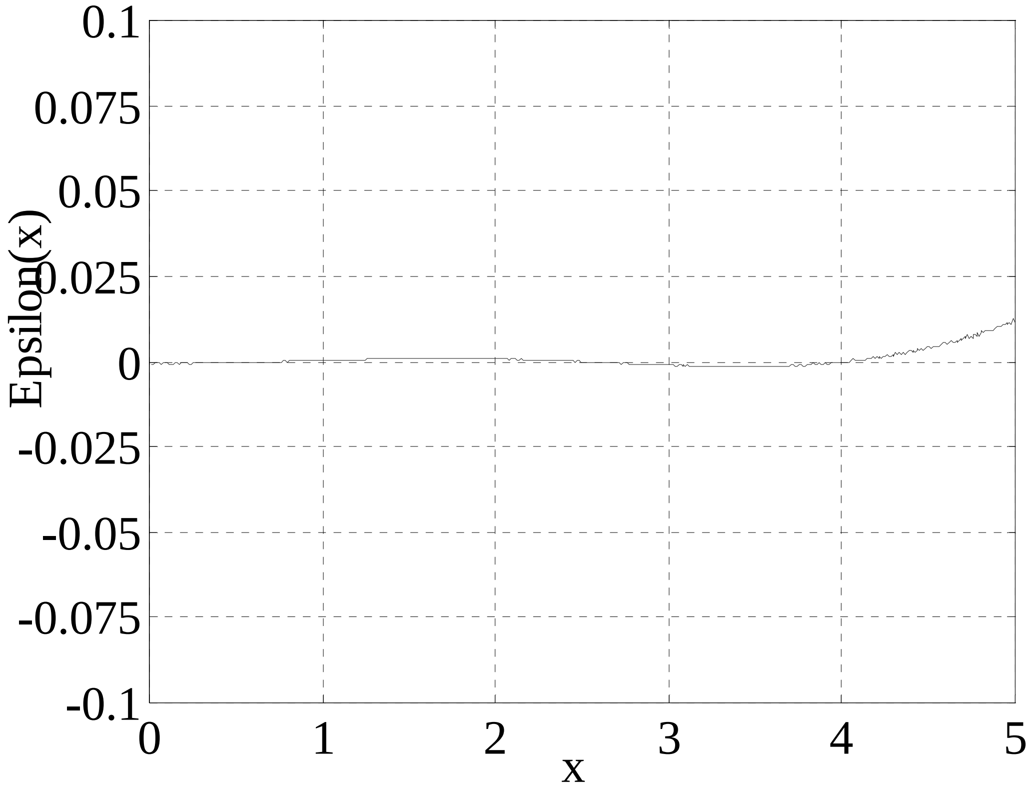


BM_2 is much better than BM_1, but still not (4σ, 1%) normal-like p.d.f ...

…thus, use central limit theorem again

Distribution BM_4





The p.d.f. of BM4 is (4σ, 1%) normal-like

Performance results

Maximum relative error $\xi_X(x)$ between the ideal gaussian distribution and BM_A

Max $\xi_X(x) * 10^{-3}$ between 0 and 4σ		A			
		2	3	4	5
b	1 δ=0.44	0.65	0.08	0.15	0.29
	2 δ=0.453	11.5	1.96	0.93	0.43
	3 δ=0.445	20.2	2.12	0.56	0.34
	4 δ=0.467	64.6	5.4	0.71	0.31
	5 δ=0.467	57.3	5.4	1.12	0.69
	6 δ=0.467	71.9	5.8	1.38	0.93
	7 δ=0.467	237	8.4	0.68	0.28
	8 δ=0.467	503	26.5	1.76	0.26

b **: number of bits after decimal point**

A : number of accumulations

***q*=4, *K*=5, *q'*=8**

The quality can be controlled with MATLAB

Outline

I Introduction

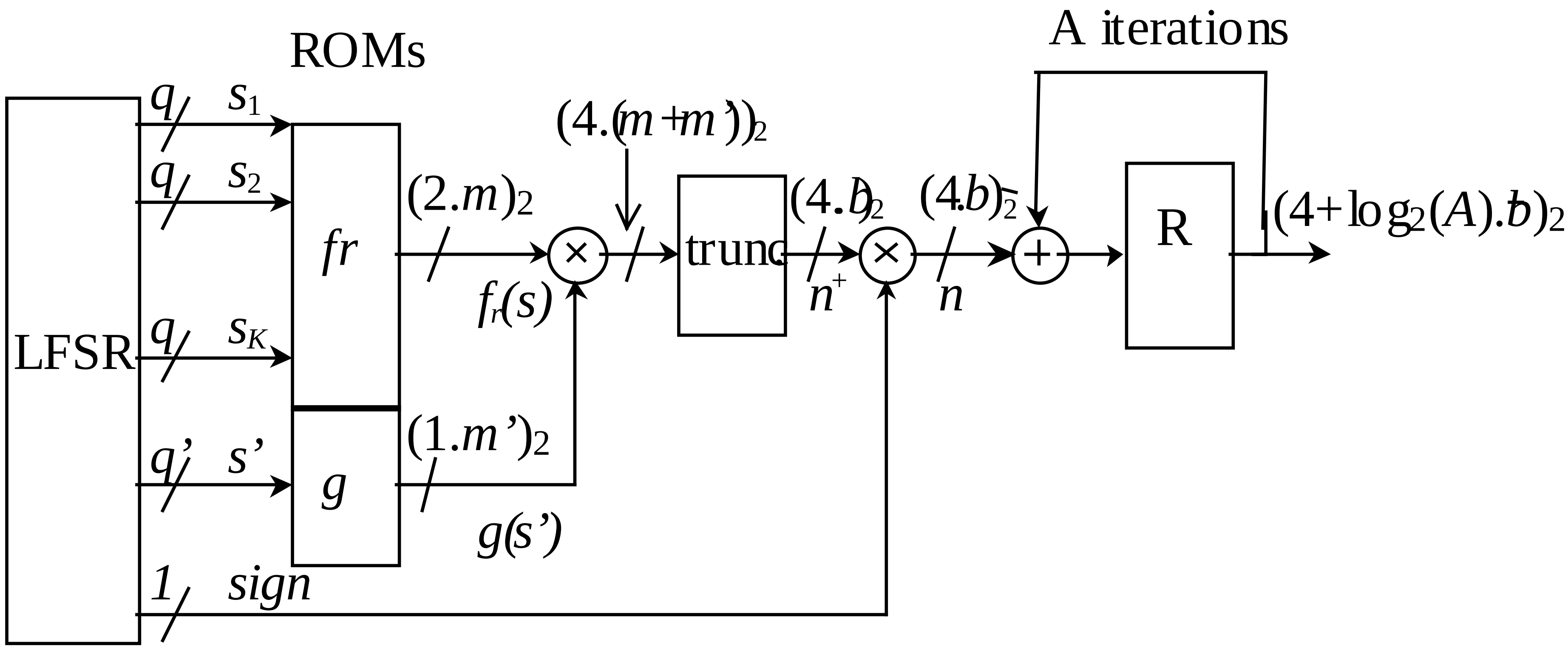
II Previous White Gaussian Noise Generator

III Proposed WGNG

IV Hardware architecture design

V Conclusion

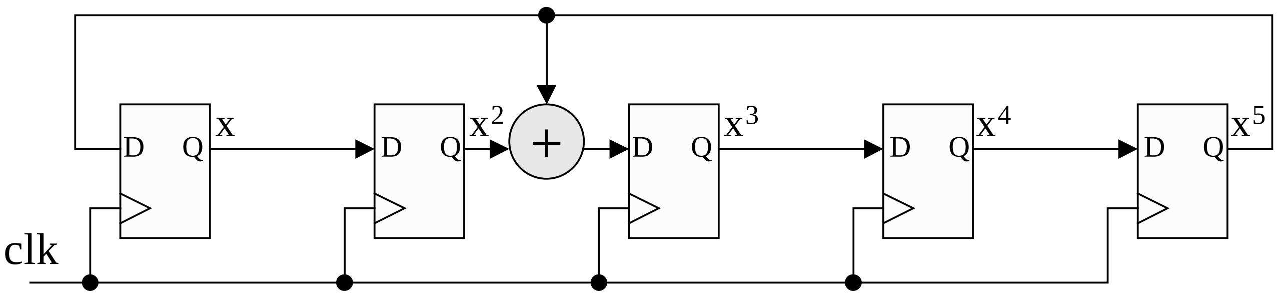
Global architecture



Architecture complexity = f(parameters)

Generation of binary variables

A LFSR of length *l* can generate a binary "random like" sequence of periodicity 2^l-1



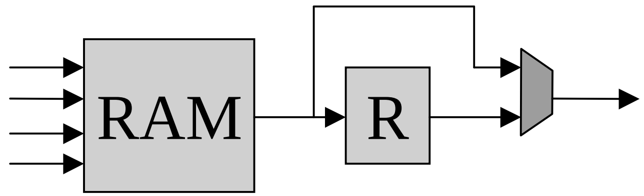
X^n mod P[X]

The periodicity of all LFSRs should be relatively prime in order to maximize the periodicity of the WGNG

=> Choice of *l* so that 2^l-1 is a prime number

Optimization for FPGA

LCELL of the FPGA:



=> q=4, in order to use LCELL for ROMs f_r

=> Use LFSR performing X^{4n} mod P[X] instead of X^n mod P[X]:
- 4 bits generated per cycle instead of 1 bit
- Same hardware complexity

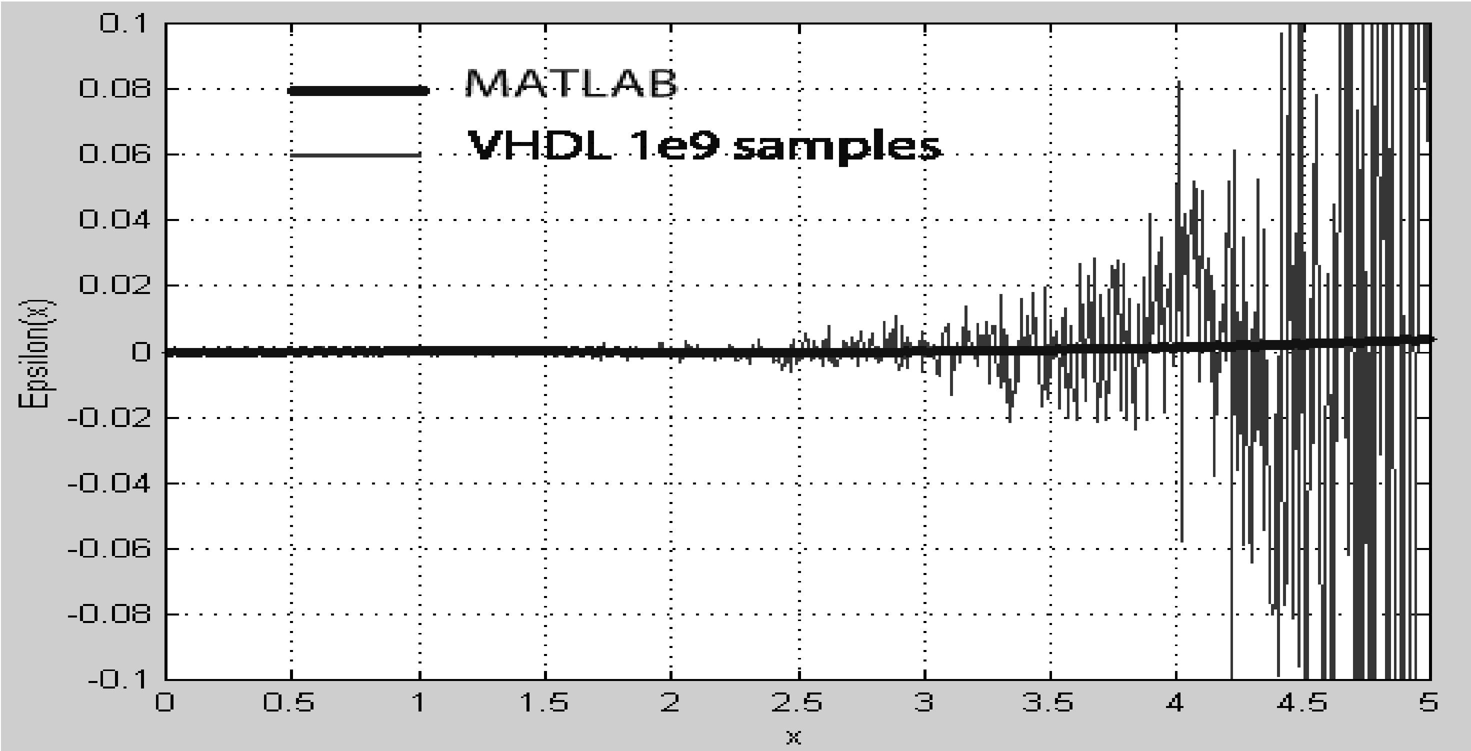
Synthesis results

Parameters:
- A = 4
- *b* = 6
- LFSR length = 22,21,20,17,13,7,15 (G, Fr and sign)

FPGA device	cells	memory block	clock rate	Output rate
10K100AR C240-1	434	1	74MHz	18.5MHz
10K100EQ C240-1	437	0.5	98MHz	24.5MHz

Less than 10% of FLEX10K100 resources

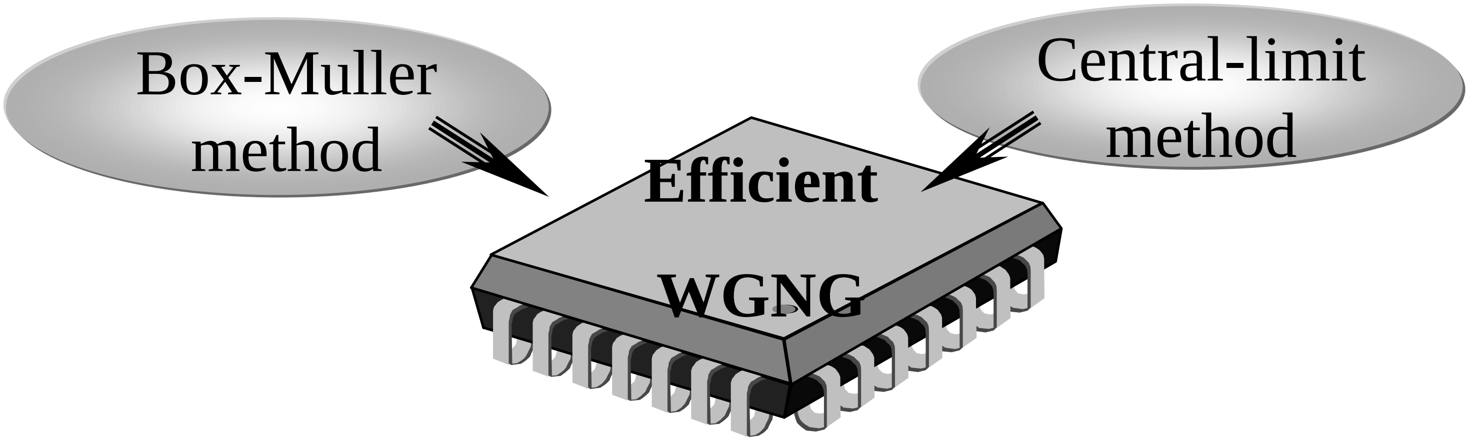
Experimental results



Theoretical distribution = measured distribution

Outline

Conclusion



Parameterizable low complexity WGNG

Quality can be fixed

Undergoing work to extend WGNG to Rayleigh Noise generator