A Fully-digital Chaos-based Random Bit Generator

M. Bucci marco.bucci@infineon.com
R. Luzzi raimondo.luzzi@infineon.com

Outline

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - Intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

- **Introduction**
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - Intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

Random and Pseudo-random Generators: different in goal, enabling technique and testing method

Pseudo-Random Bit Generator	Random Bit Generator
Goal	
Uniform distribution: generating data that **look** random	Maximal entropy: generating data that **are** random
Enabling technique	
Cryptography: deterministic finite state machines using strong one-way functions	• Noise/Entropy generation • Protection against disturbances • Entropy extraction • Entropy concentration
Testing Method	
(deceiving) Statistical hypothesis test: can we hide the fact that there is no entropy?	Entropy evaluation: are we close to the maximal entropy density?

Entropy evaluation vs classes of entropy sources

1. **Noise model not available (or technology dependent)**

2. **Noise model available (and technology independent)**

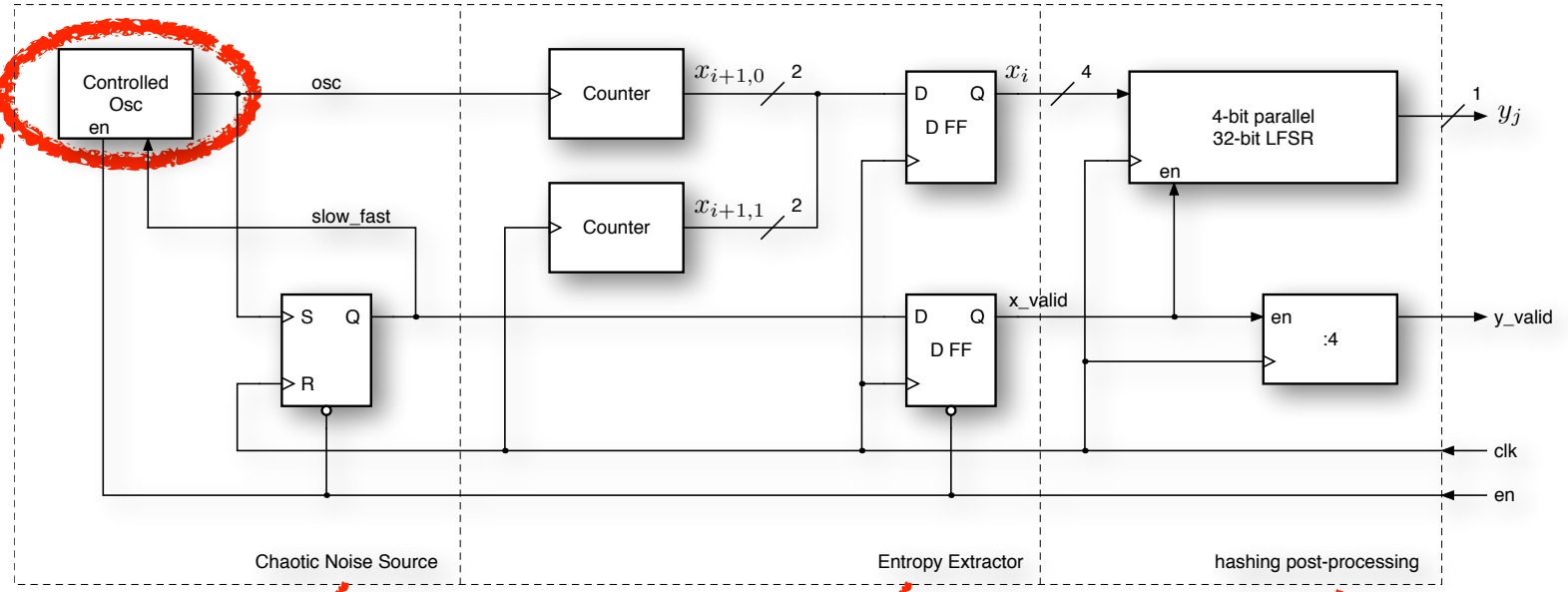
3. **Noise model does not matter (chaotic sources)**

   Entropy rate depends only on the Lyapunov exponent of the system.
   Any other parameter non affecting the Lyapunov exponent could eventually change the statistical distribution, but not the entropy rate.

Assessing entropy by means of statistical distribution (of noise or of symbols) is **not only useless, but also misleading**.

- Introduction
- **Implementation overview**
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - Intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

Implementation overview



Two-speed (F_{fast}/F_{slow}) ring oscillator

Constant entropy rate: $\log_2(F_{fast}/F_{slow})$

Full entropy extraction by means of a **generating partition**.

Full entropy output.

Testable by means of a suitable predictor.

Expected operating frequencies and entropy rate (transistor level simulation)

Worst case assumptions:

- Noise free VDD = 1.25V
- Jitter free CLK oscillator

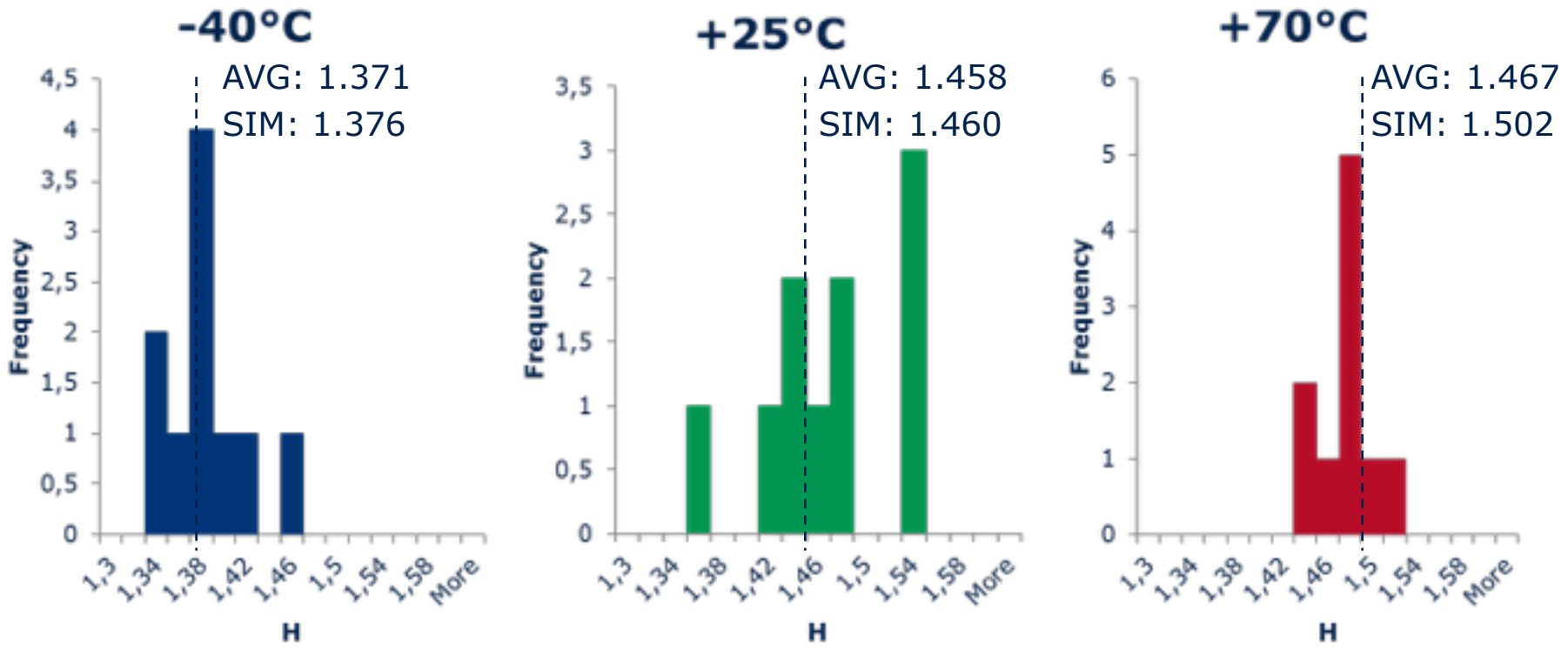
Technology: CMOS 40nm

	Unit	-40°C	+25°C	+70°C
Fast frequency	MHz	421	381	357
Slow frequency	MHz	160	139	126
Ratio	-	2.637	2.752	2.832
H rate = $\log_2$(Ratio)	bits	1.399	1.460	1.502

Estimated entropy vs. Temperature (real data on 10 nominal process chips)

Conditions:

- 160Mbit raw data
- 10 NOM chips, -40°C, +25°C,+70°C

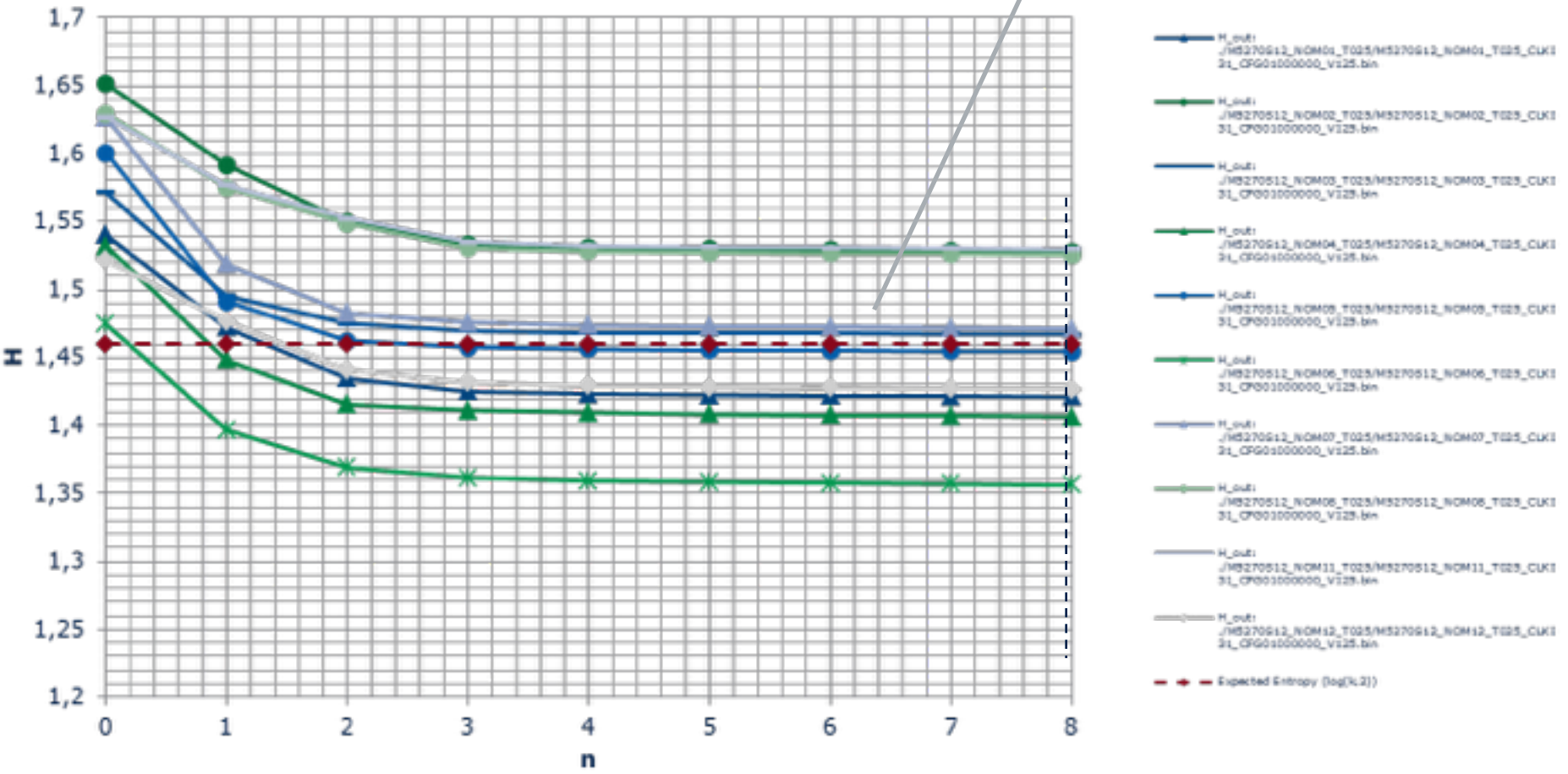


Conditional entropy estimation (real data on 10 nominal process chips)

Conditions:

- 160Mbit raw data
- 10 NOM chips, +25°C

- Measured H fits to the model
- No long-term dependencies



- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - **Summary of known results**
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - Intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

Well known results on chaotic systems

Entropy rate

- the system has an “intrinsic” (Shannon) entropy rate which is defined by the Kolmogorov-Sinai entropy
- the intrinsic **entropy rate is equal to the Lyapunov exponent** (Yakov B. Pesin) (i.e. the entropy rate does not depend on the noise model)

Entropy extraction

- the full entropy rate of the system can be extracted by using a **generating partition** of the phase space

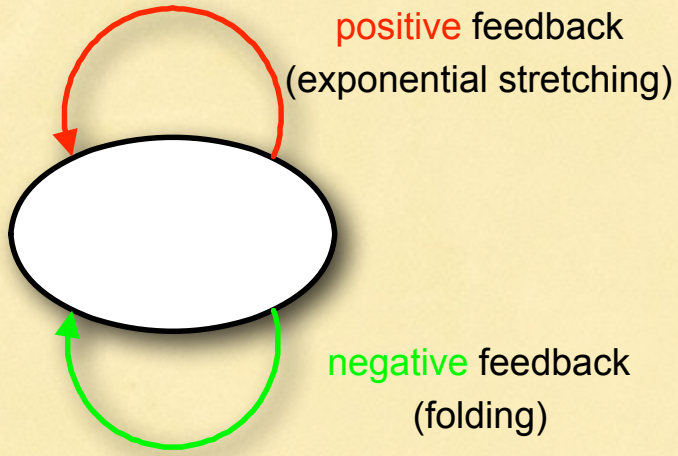
Entropy evaluation feasibility

- memory (i.e. statistical dependency) decreases exponentially
- system is **mixing** ⇒ **ergodic** ⇒ **stationary**

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - **Bernoulli map and generalised Sawtooth map**
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

Chaos Based Sources: Principle of Operation

Chaotic systems can be seen as a "**stretching and folding**" mechanism.

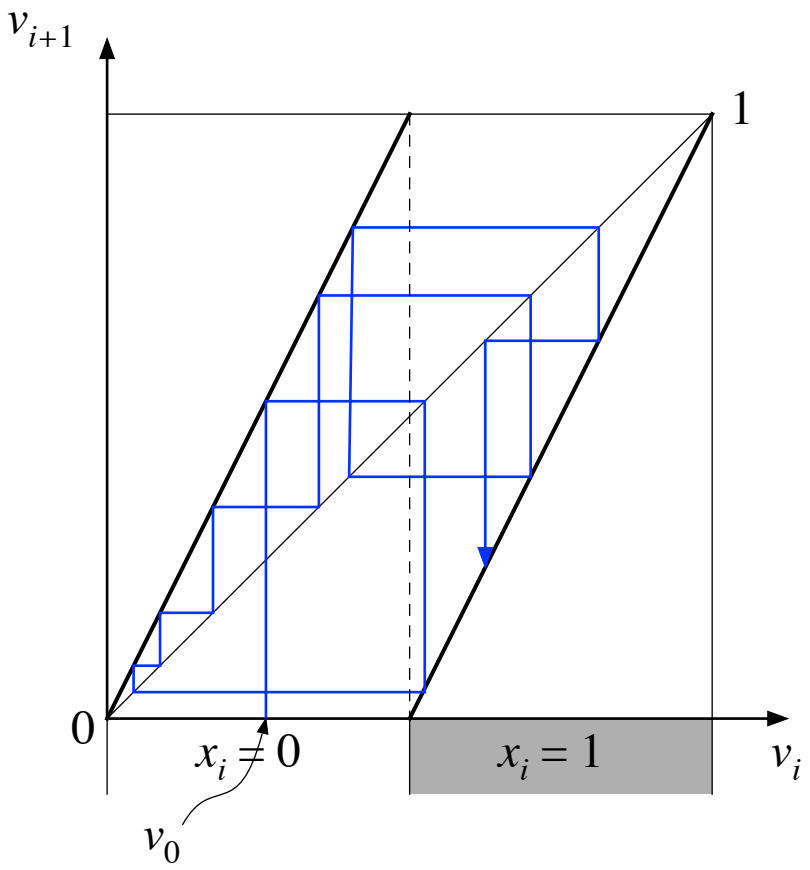


- a positive feedback loop exponentially stretches (amplifies) state variables
- a negative, non linear, feedback loop folds (constraints) the state evolution inside the dynamic range of the system

With respect of traditional solutions, **noise amplification and external disturbances are not anymore an issue** (both, noise and disturbances, are exponentially amplified and the effect cannot be controlled by an attacker).

The main issue is finding a robust implementation since, **If one of the two loops prevails (positive vs negative), the system gets saturated or switches off**.

Bernoulli map: an ideal binary entropy source



Discrete time chaotic system:

$$v_{i+1} = \mathrm{mod}\,(2 \cdot v_i, 1)$$

folding — exponential stretching

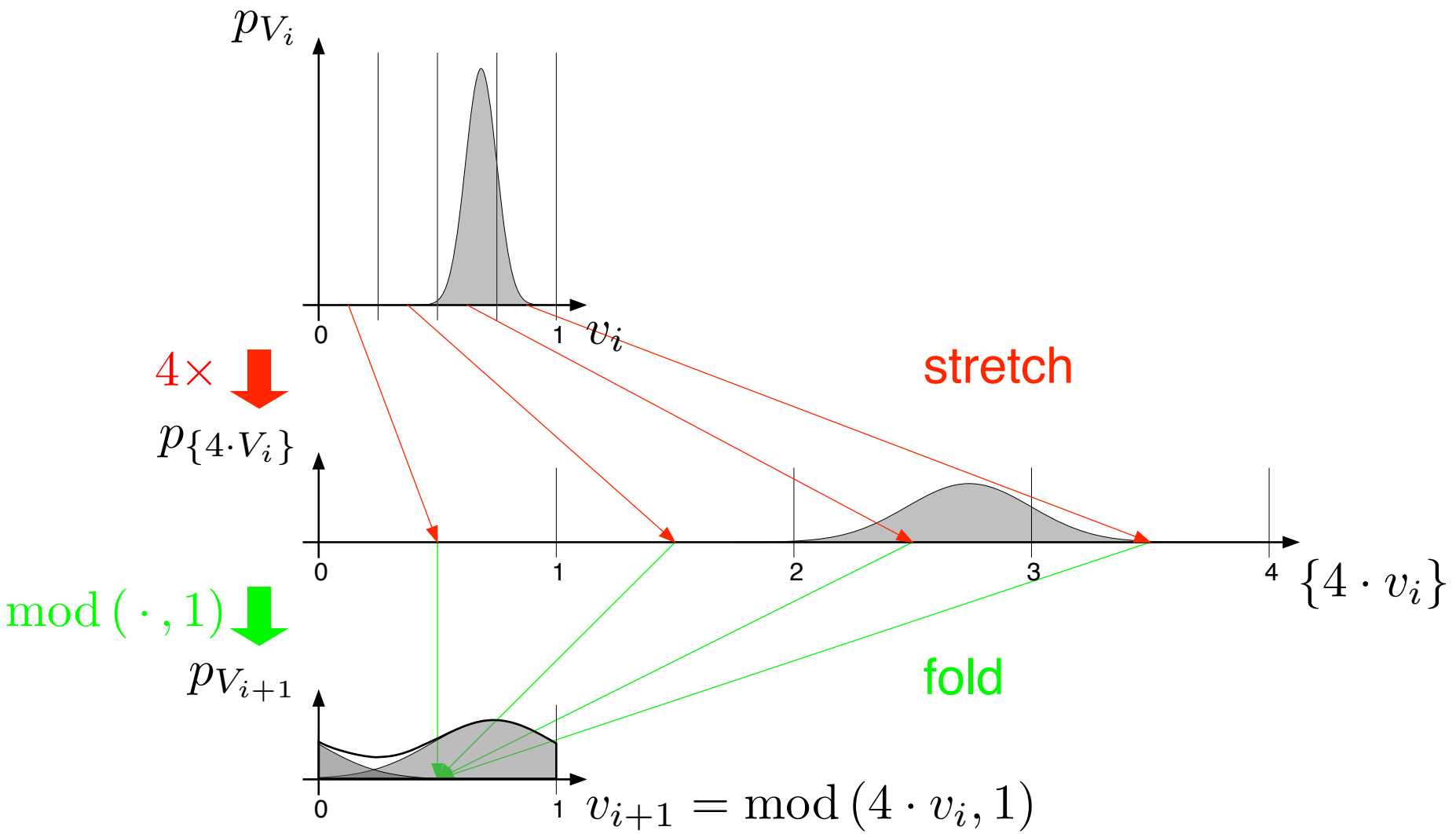
Whichever $\rho(v_0)$ is, $\rho(v_i)$ converges to $\rho^{inv}(v) = 1$ exponentially fast.

The generated sequence $x_i = \lfloor 2 \cdot v_i \rfloor$ actually consists of the binary representation of the initial state:

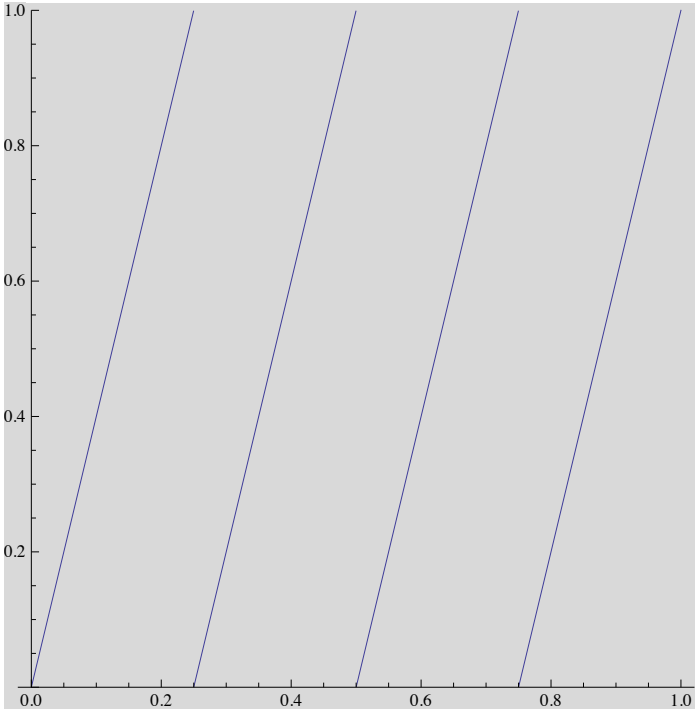
$$v_0 = \sum_{i=0}^{\infty} x_i \cdot 2^{-(i+1)}$$

and, it can be seen, it has **maximal entropy**.

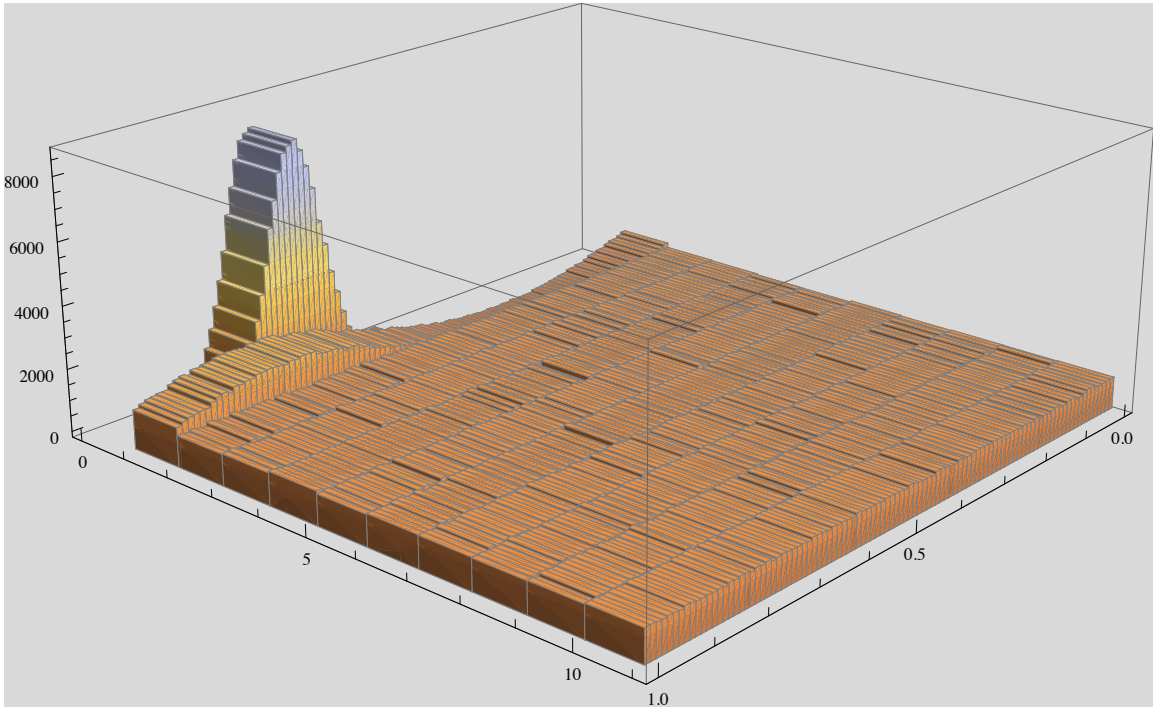
Stretching and folding (e.g. $v_{i+1} = \mathrm{mod}(k \cdot v_i,\ 1);\ k = 4$): how the noise amplification operates



Convergency to the invariant distribution for k integer (e.g. $k = 4$)



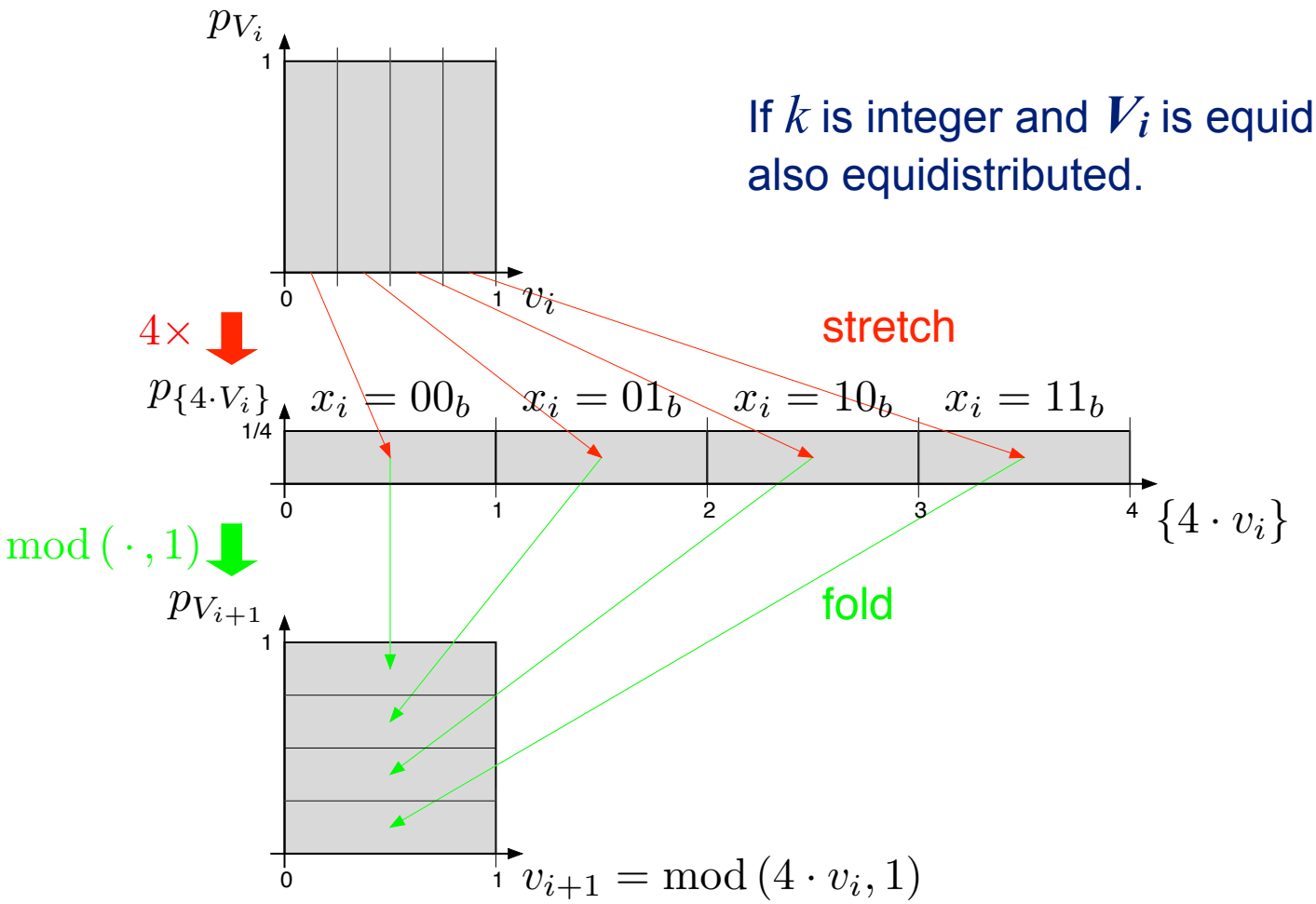
chaotic map



Evolution of the state distribution vs time step: convergency to the invariant distribution

It can be proven that, whichever is the initial distribution, the system state exponentially converges to an **invariant** (i.e. stationary) distribution that, since k is integer, is also uniform.

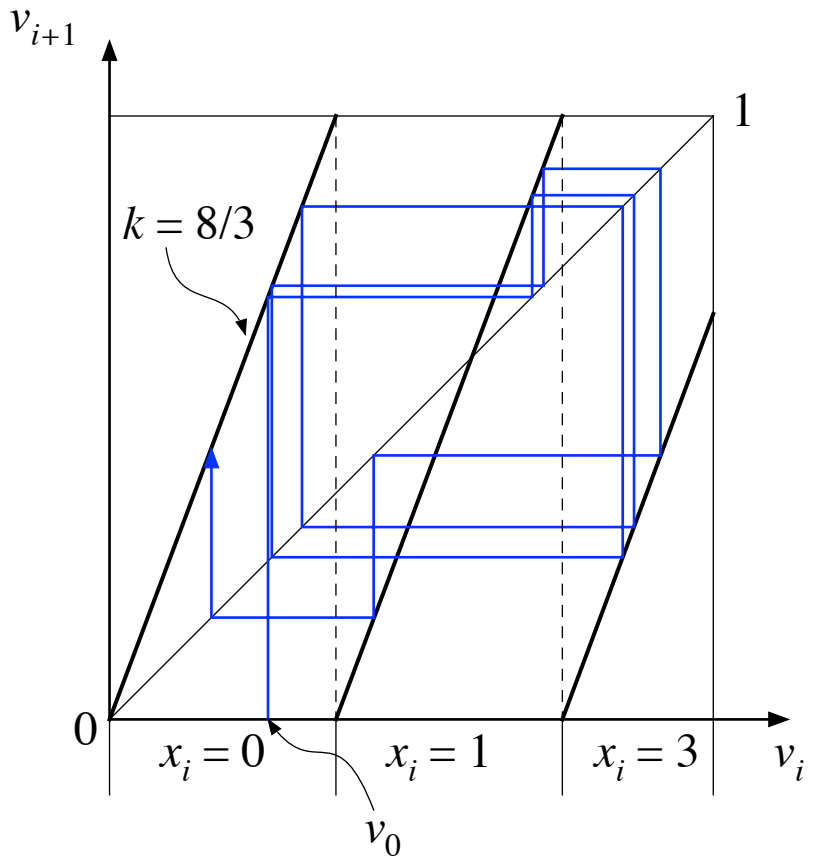
Mutual information and memory for k integer (e.g. k = 4)



If k is integer and V_i is equidistributed, V_{i+1} is also equidistributed.

There is no mutual information between V and X and hence no mutual information (i.e. no memory) between x_i and x_{i+1}.

Generalised Sawtooth Map



Generalisation of the Bernoulli map:

$$v_{i+1} = G(v_i) = \mathrm{mod}(k \cdot v_i, 1)$$

$$x_i = C(v_i) = \lfloor |k| \cdot v_i \rfloor$$

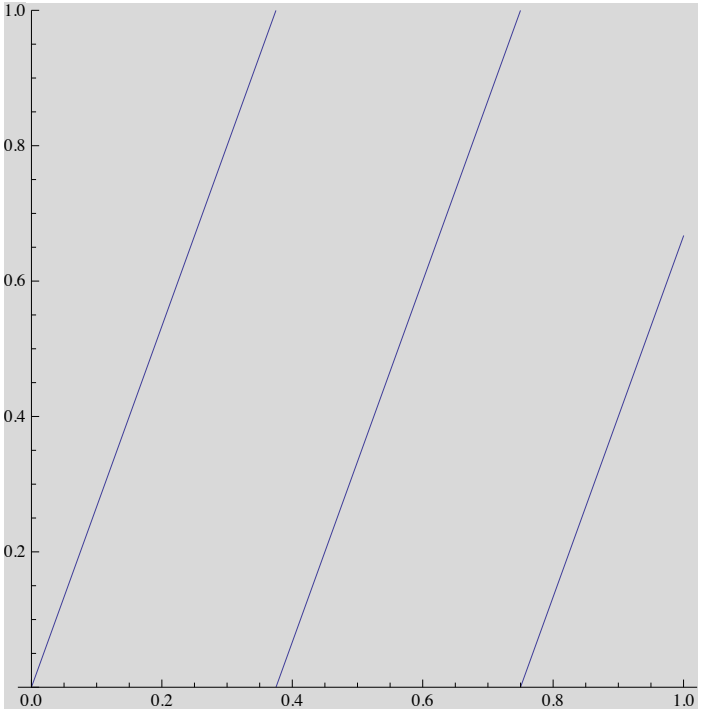
$$|k| > 1$$

If k is not integer, $\rho^{inv}(v)$ is not uniform and the generated sequence is not maximal entropy (symbols are not equidistributed and not independent).

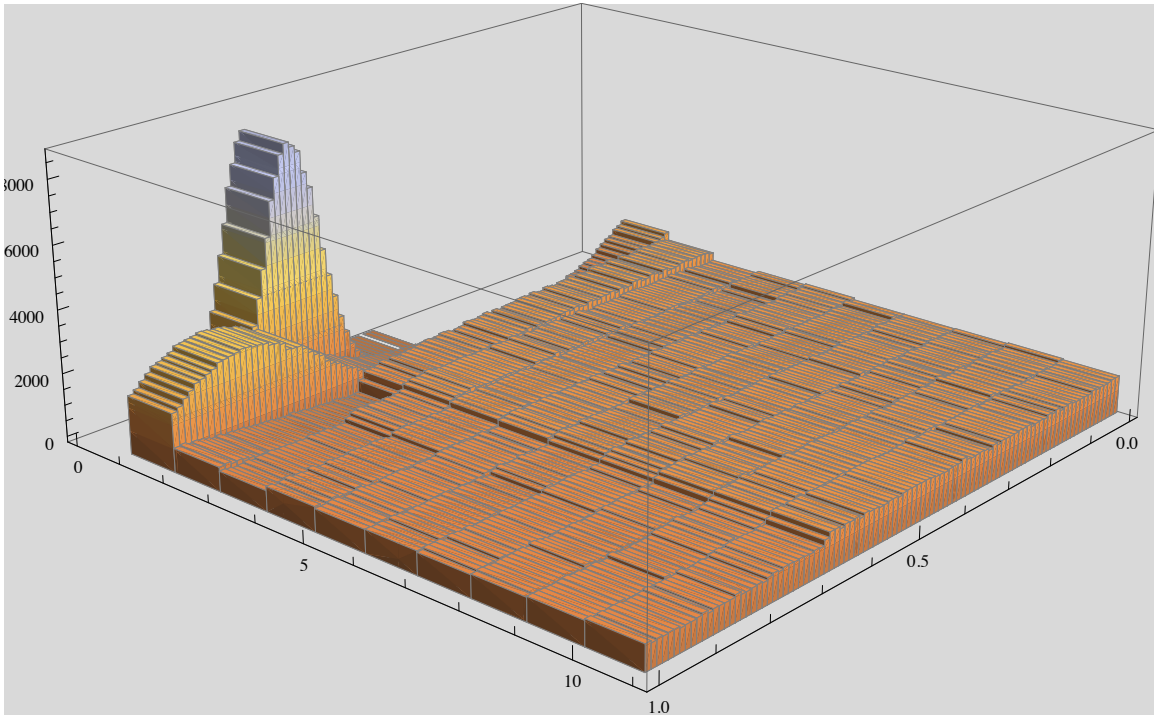
Nevertheless the **entropy rate depends only on the Lyapunov exponent of the system** and it holds:

$$h(X) = \log_2 |k|$$

Convergency to the invariant distribution for k non integer (e.g. k = 8/3)



chaotic map



Evolution of the state distribution vs time step: convergency to the invariant distribution

It can be proven that, whichever is the initial distribution, the system state exponentially converges to an **invariant** (i.e. stationery) distribution but, since k is not integer, is not uniform. Moreover, the successive values of the state are not independent,

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - **Why the noise model does not matter**
  - Proposed chaotic system
- Entropy extraction
  - Intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

"Noisy" Discrete time system representing a **chaotic**, a **free−running** or a **PLL** oscillator depending on *k*

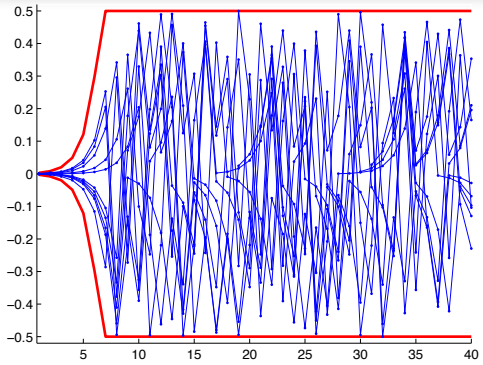
1-dimensional state map: $\bar{v}_{i+1} = \mathrm{mod}\left(k \cdot \bar{v}_i + 0.5 + \eta_i, 1\right) - 0.5$

state envelop: $b_i = \pm \min\left(0.5, 3 \cdot \sigma_{v_i}\right) = \pm \min\left(0.5, 3 \cdot \sqrt{\sum_{j=0}^{i} \left(k^{i-j} \cdot \sigma_\eta\right)^2}\right)$

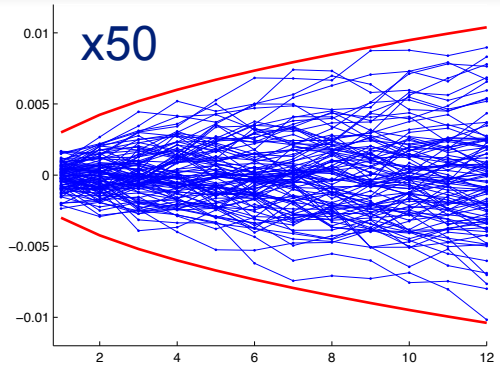
$\bar{v}_i$ —

b_i —

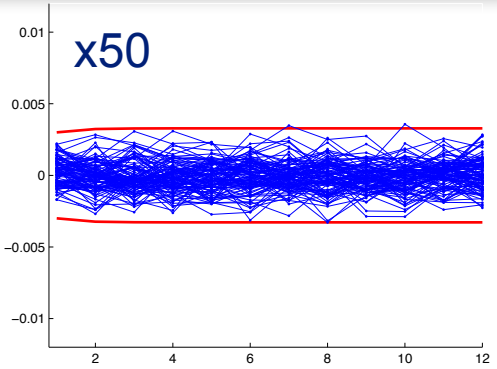
$\sigma_\eta = 10^{-3}$



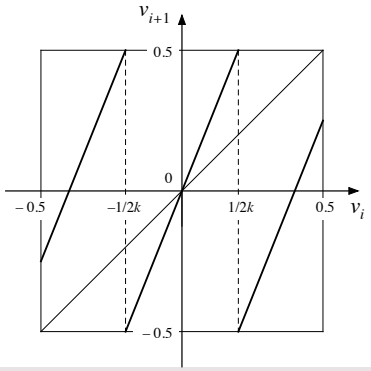
$k = 2^{1.3} > 1 \Rightarrow$ Chaotic

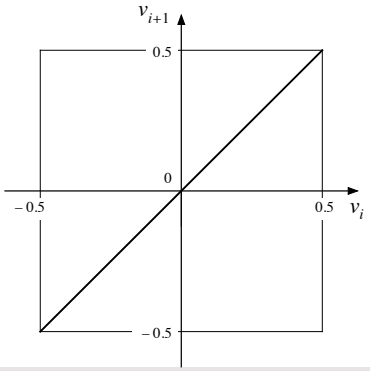


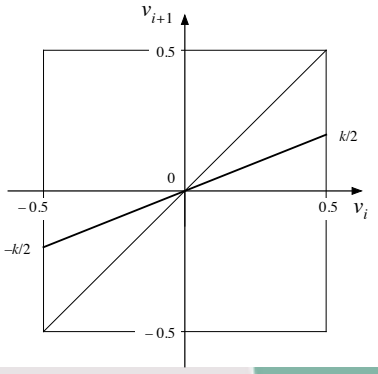
$k = 1 \Rightarrow$ Free Running



$k = 2^{-1.3} < 1 \Rightarrow$ PLL

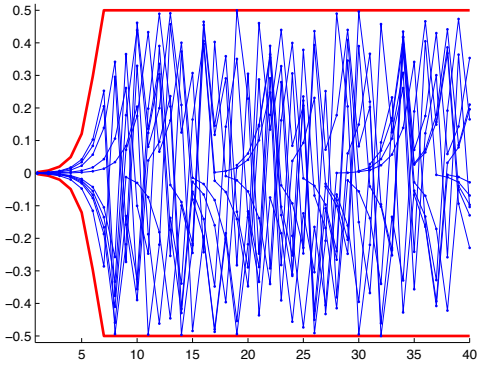




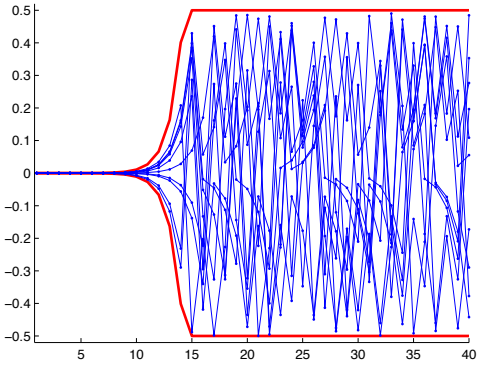


Noise intensity does not practically matter

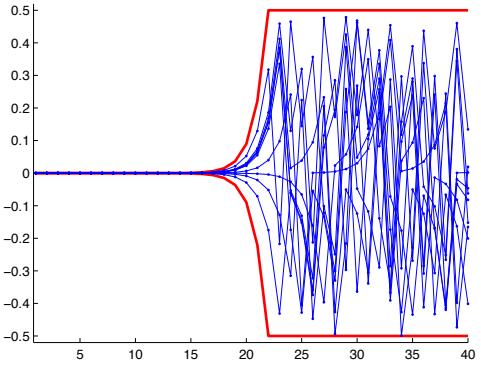
$k = 2^{1.3}$



$\sigma_\eta = 10^{-3}$



$\sigma_\eta = 10^{-6}$



$\sigma_\eta = 10^{-9}$

The time needed to reach the steady state depends just logarithmically on the noise intensity.

However, in steady state, the system behaviour does not depend on noise intensity.

Entropy rate is practically independent from noise intensity

Using a suitable **generating partition**, the “intrinsic” entropy rate of the system can be estimated:

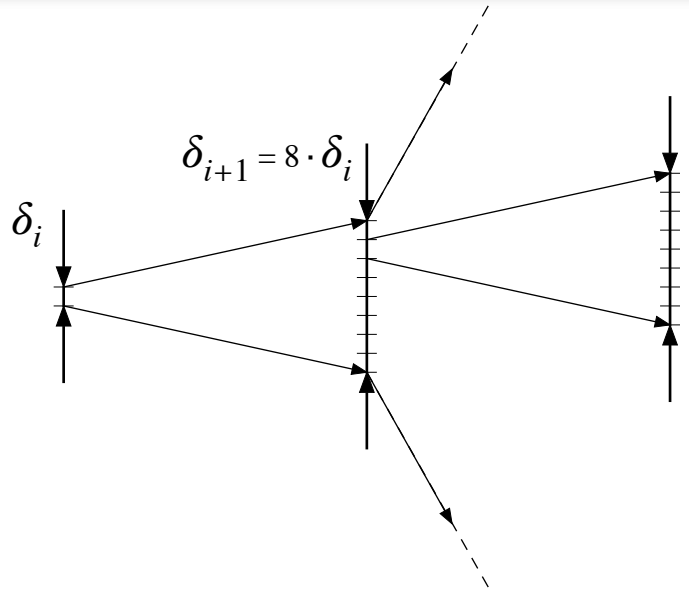
noise intensity has some effect only when the system is very “weakly” chaotic.

k	σ_η 10^{-3}	10^{-6}	10^{-9}	$H(X) = \log_2 k$
$2^{0.1} \approx 1.071$	0.463	0.107	0.107	0.1
$2^{0.2} \approx 1.148$	0.318	0.202	0.202	0.2
$2^{0.3} \approx 1.231$	0.337	0.300	0.300	0.3
$2^{0.4} \approx 1.319$	0.417	0.400	0.400	0.4
$2^{0.5} \approx 1.414$	0.507	0.499	0.499	0.5
$2^{0.6} \approx 1.515$	0.603	0.598	0.598	0.6
		$\hat{H}(X)$		

Estimated entropy rate $\hat{H}(X)$ vs noise intensity σ_η, multiplication factor k and the expected entropy rate $H(X) = \log_2 k$.

Why entropy rate coincides with Lyapunov exponent?

In a chaotic system the uncertainness of the observer regarding the state of the system grows exponentially according the Lyapunov exponent λ: E.g. in a 1-dimensional discrete time system: $\delta_{i+1} \approx \delta_i e^{\lambda}$.

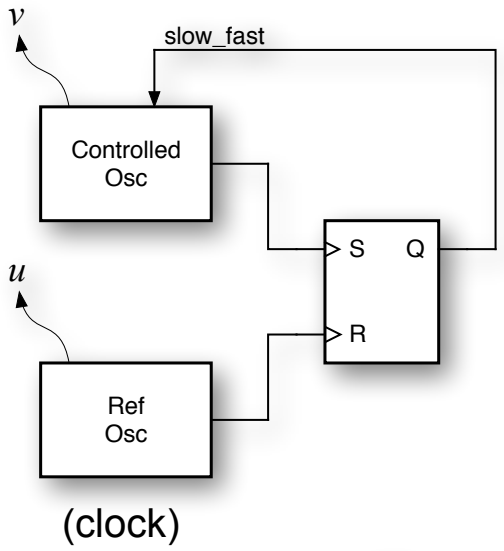


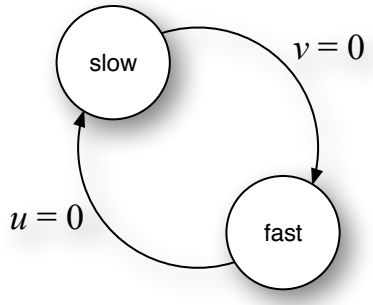
The entropy rate of the system coincides with the information rate that an observer should get, at each step, in order to trace the state of the system maintaining the same uncertainness (e.g. 3 bits if $e^{\lambda} = 8$).

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - **Proposed chaotic system**
- Entropy extraction
  - Intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

Proposed noise source: continuos time representation

The proposed chaotic system is implemented in the time domain where **the modulo operation is implicit in the nature of the state variables** (oscillator phases).





phase of "reference oscillator":

$$u(t) = \mathrm{mod}(t, 1)$$

phase of "controlled oscillator":

$$\lim_{\Delta t \to 0} v(t + \Delta t) = \mathrm{mod}(v(t) + \dot{v}(t) \cdot \Delta t, 1)$$

"controlled oscillator" has two speeds:

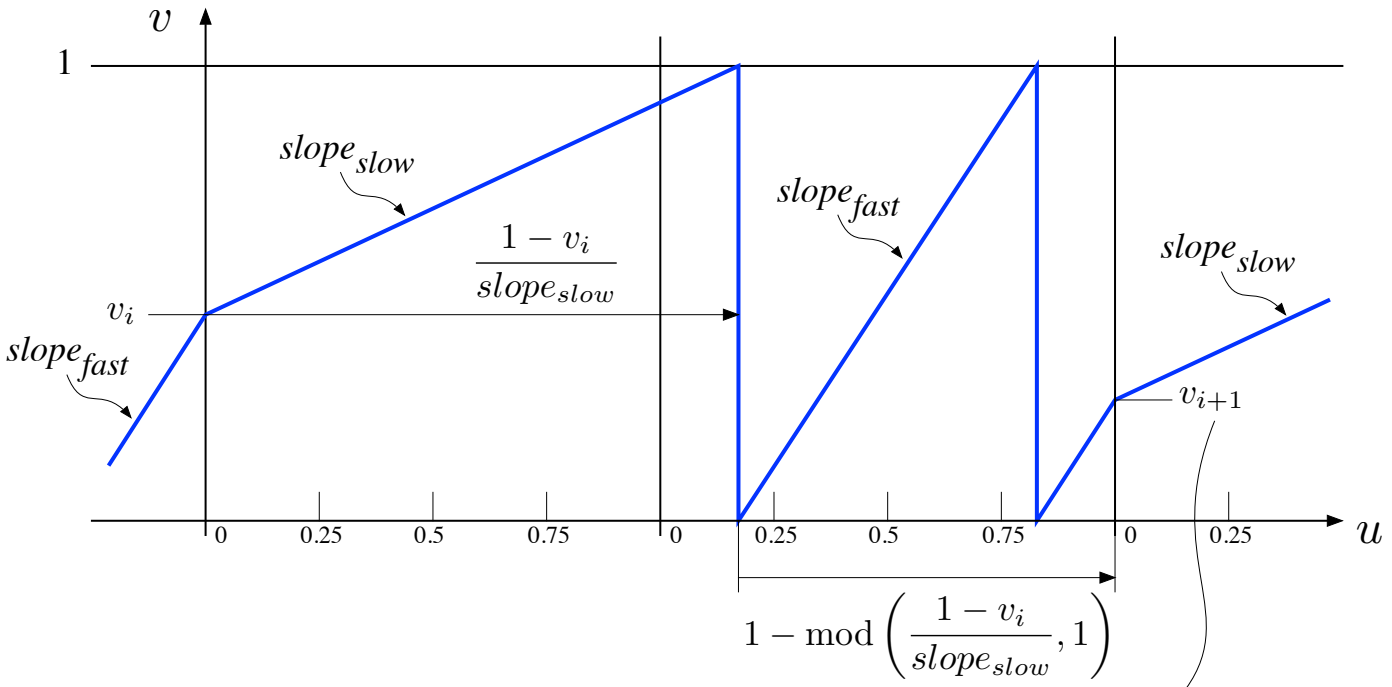
$$\dot{v}(t) = \begin{cases} slope_{slow} & \text{if } mode = slow \\ slope_{fast} & \text{if } mode = fast \end{cases}$$

speed control logic:

$$mode \leftarrow \begin{cases} slow & \text{when } u = 0 \\ fast & \text{when } v = 0 \end{cases}$$

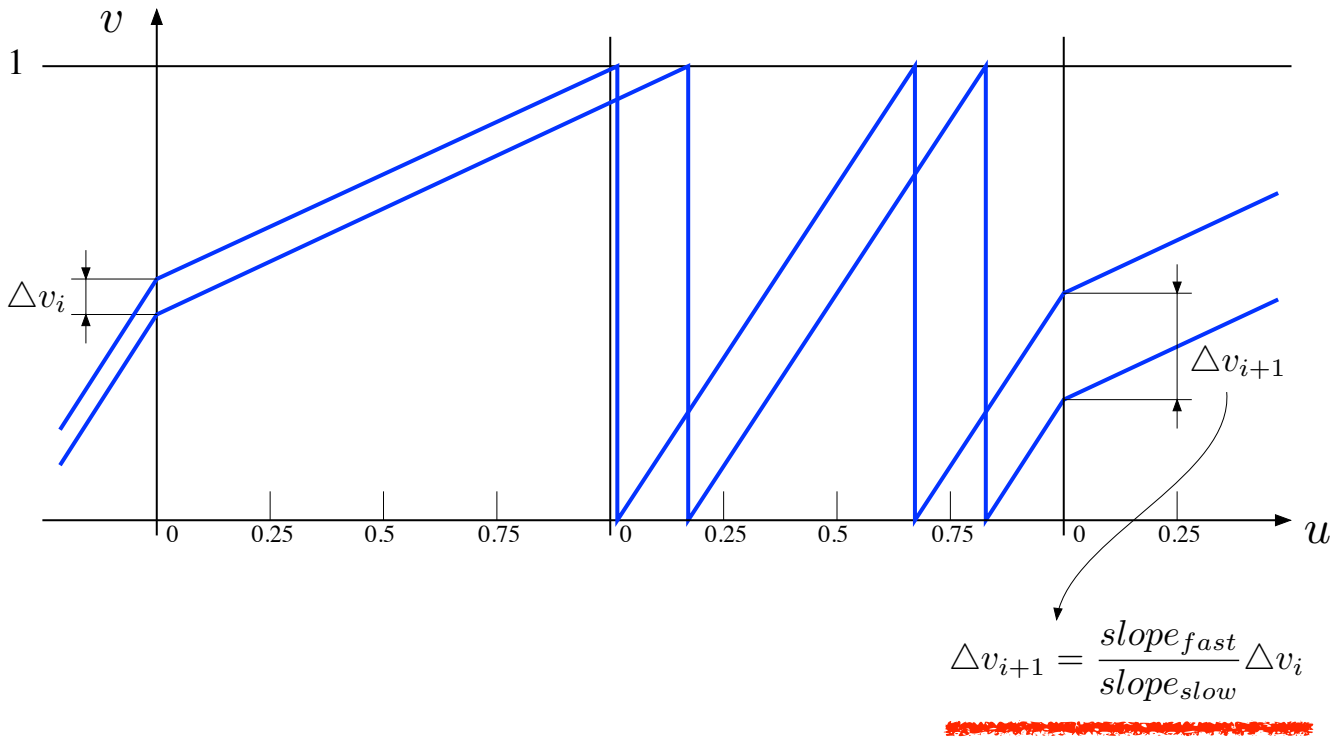
Proposed entropy source: discrete time representation

By defining a system iteration as the period between two fast -> slow transitions, it is possible to define the chaotic map (i.e. the discrete time representation).



$$v_{i+1} = \mathrm{mod}\left(slope_{fast}\left(1 - \mathrm{mod}\left(\frac{1-v_i}{slope_{slow}}, 1\right)\right), 1\right)$$

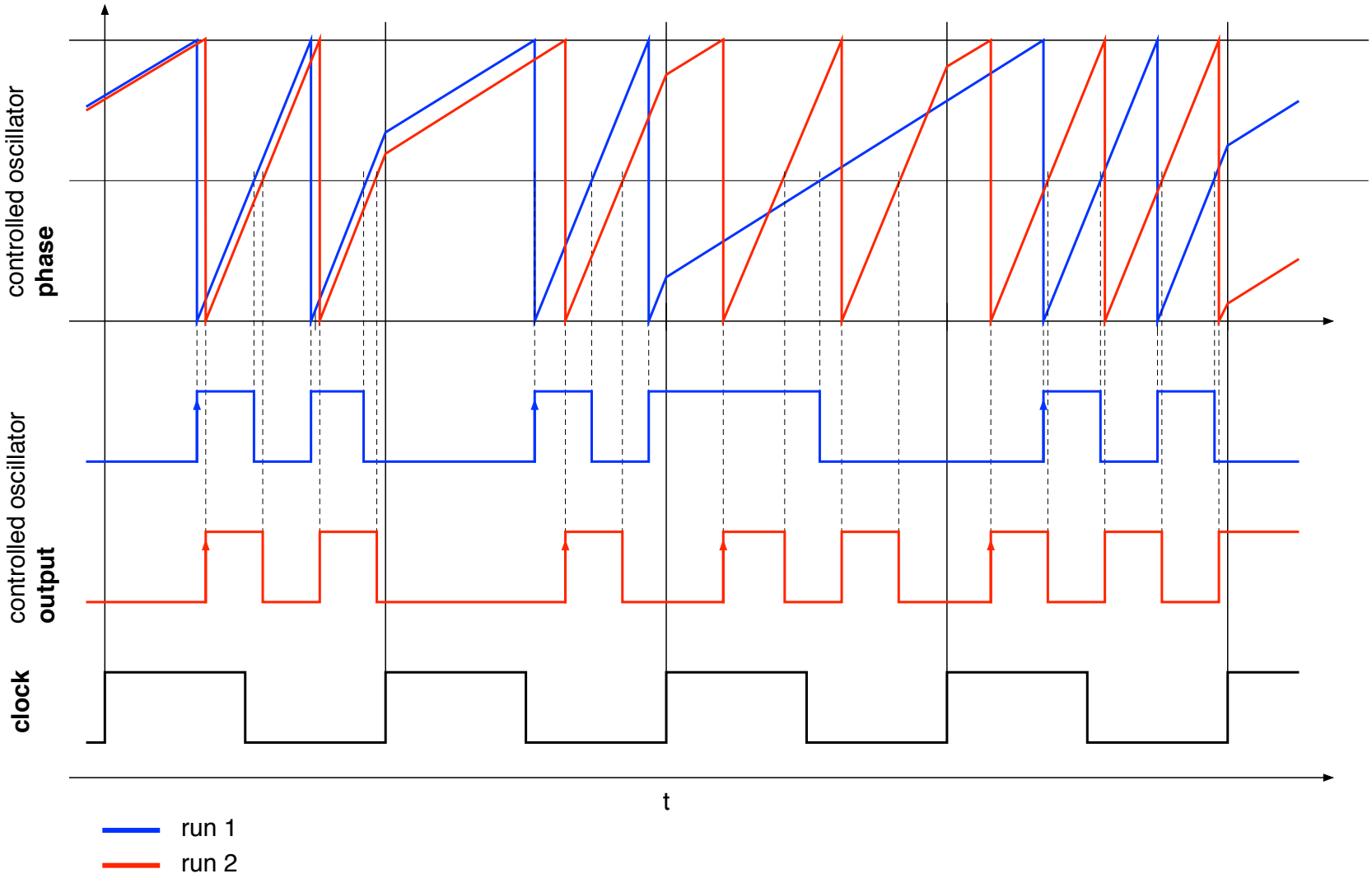
Evolution of two trajectories (jitter amplification mechanism)



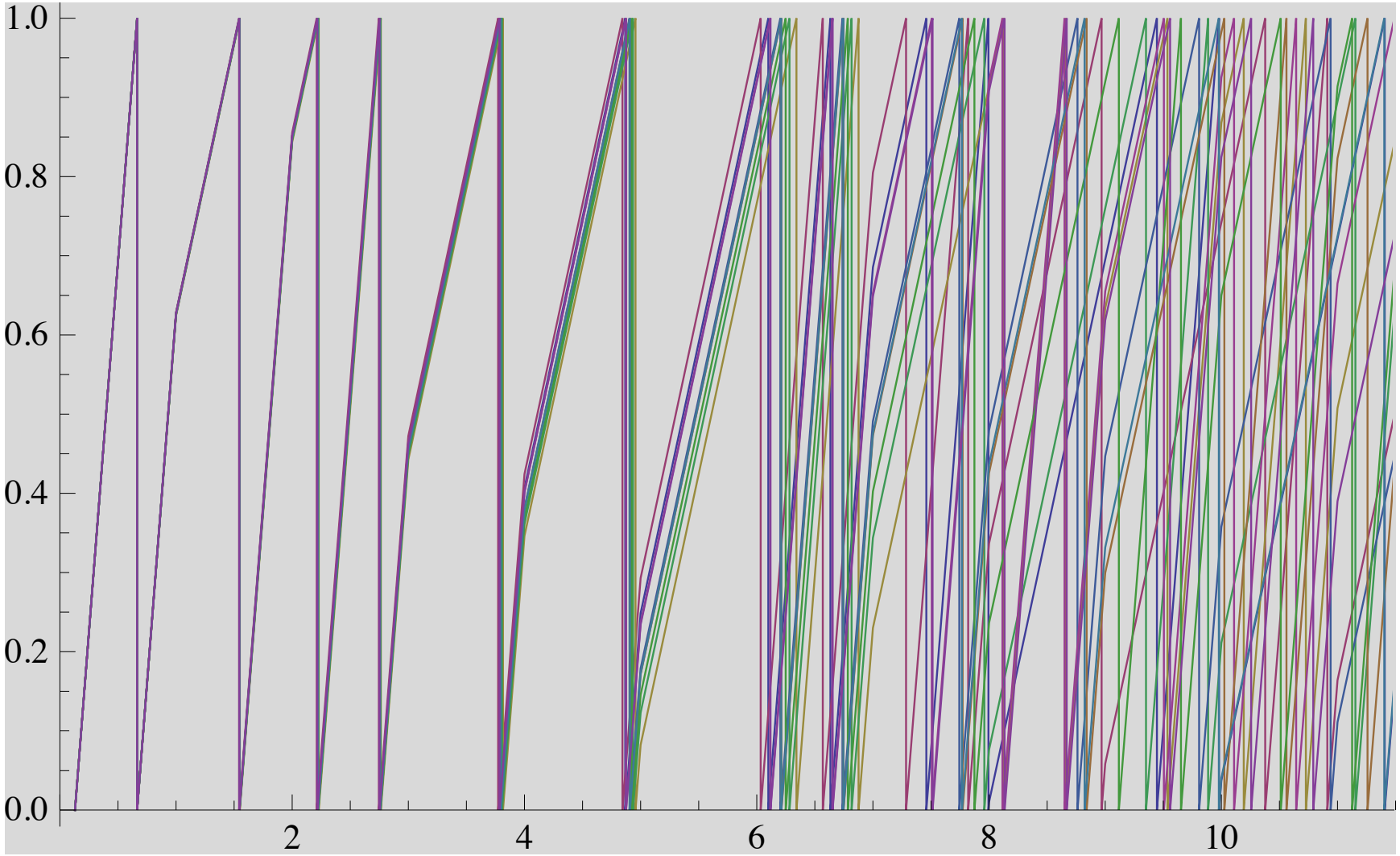
Noise (i.e. jitter) amplification results from the separation between trajectories which follows the law $|\delta v_{i+n}| = |\delta v_i|\, k^n$ where $k = slope_{fast}/slope_{slow}$.

This is the evidence that the Lyapunov exponent is $\lambda = \ln|k|$.

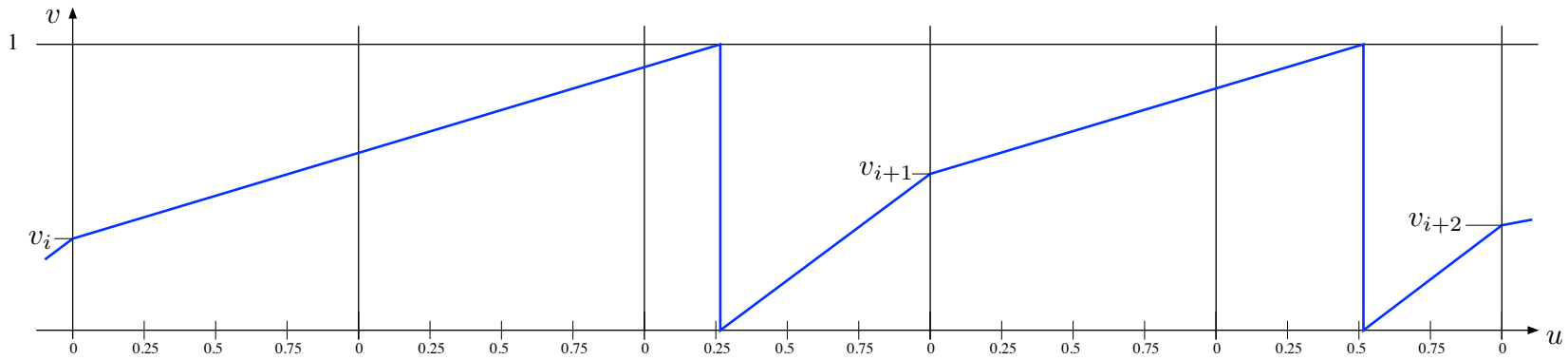
Controlled oscillator and clock traces over two runs



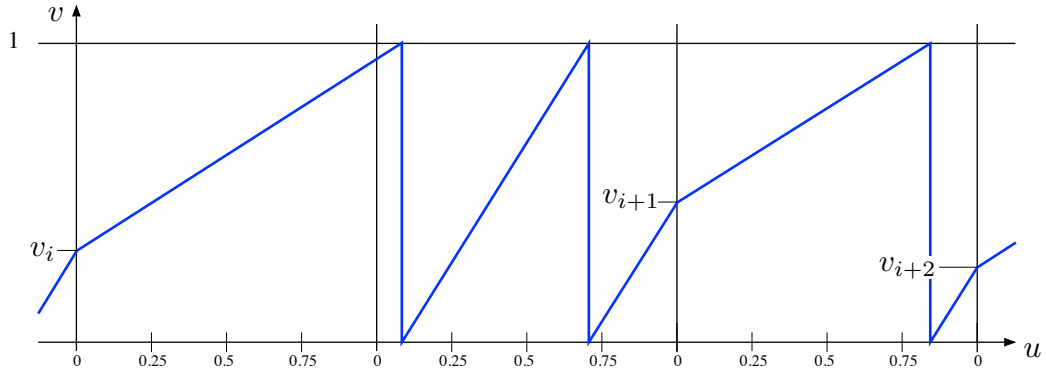
Noise amplification on 10 simulation runs
k = slopeRatio = 2.735; slopeFast = 1.87; nSigma = .001



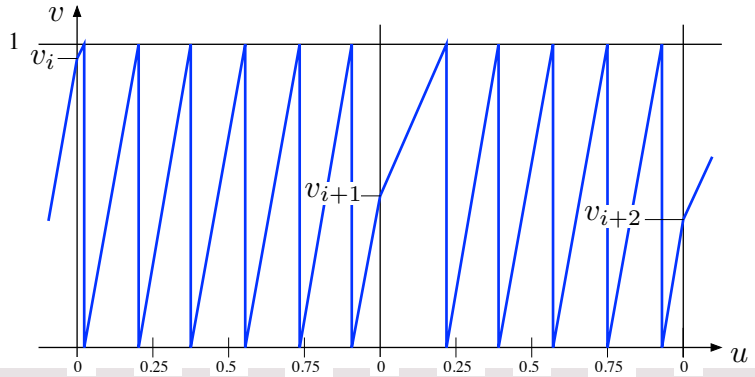
Time evolution for different clock vs controlled-oscillator frequencies



$slope_{fast} < 1$

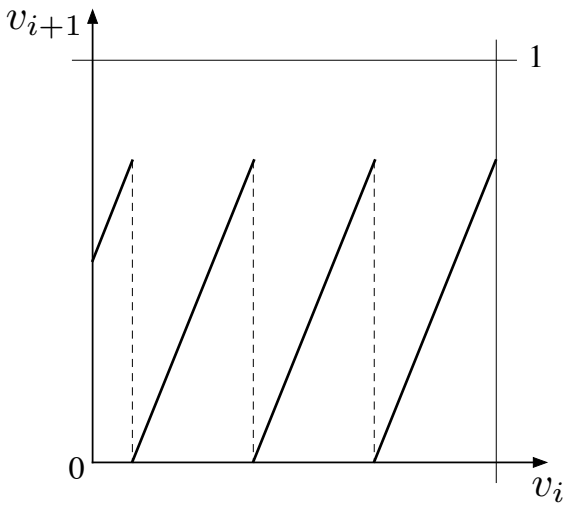


$slope_{fast} > 1$ and $slope_{slow} < 1$

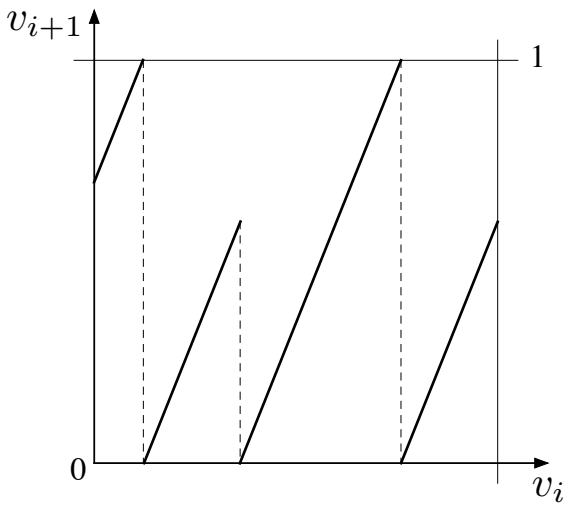


$slope_{slow} > 1$

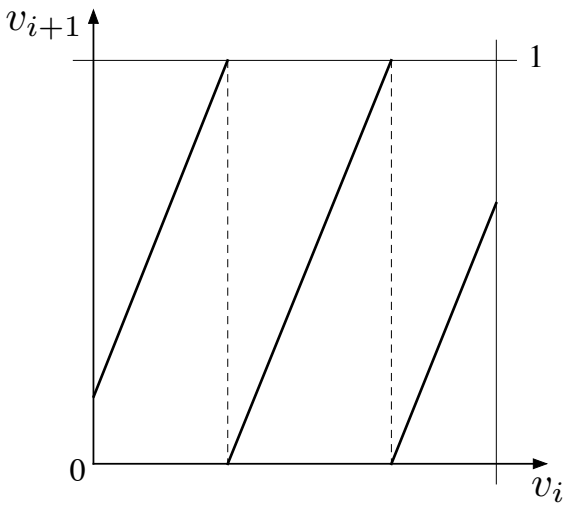
Chaotic map for different clock vs controlled-oscillator frequencies



$slope_{fast} < 1$



$slope_{fast} > 1$ and $slope_{slow} < 1$



$slope_{slow} > 1$

The chaotic map can assume three different shapes, but it is always a **piecewise linear map having constant derivative** $k = slope_{fast}/slope_{slow}$ and therefore a constant entropy rate $\log_2 |k|$.

$k = slope_{fast}/slope_{slow}$ **is the only relevant parameter of the system**.

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- **Entropy extraction**
  - Intuitive operation description
  - Formal operation description
- Post-processing and output entropy evaluation

Entropy extraction

Entropy extraction consists of the transfer of the entropy of the system (in this case the jitter of the controlled oscillator) to a digital sequence X.

FACT: the entropy rate of the system is $\log_2(k)$, i.e. it depends on a single parameter (namely the ratio between fast and slow frequency of the controlled oscillator).

GOAL: the entropy extraction mechanism should extract the full entropy rate of the system regardless of any system parameter (namely the absolute frequencies of the two oscillators)

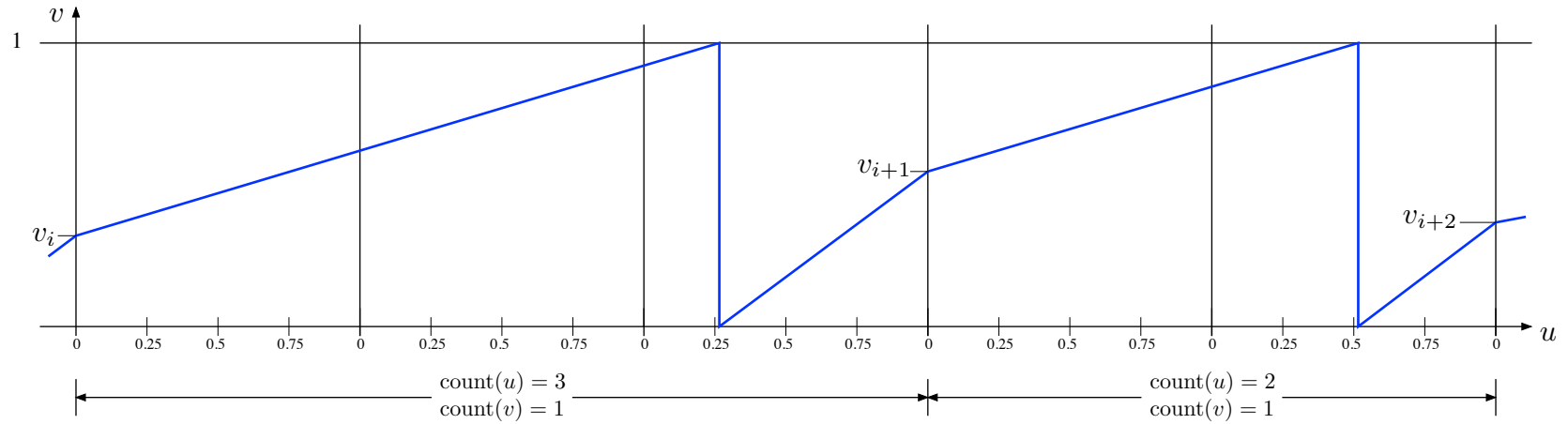
Notice: in general, X does not have maximal entropy per bit. Entropy concentration is performed by post-processing hashing.

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - **Intuitive operation description**
  - Formal operation description
- Post-processing and output entropy evaluation

Entropy extraction: slow case ($slope_{fast} \leq 1$)

The controlled oscillator ***v*** is always (i.e. also in fast mode) slower than the reference oscillator ***u***.

System iterations are always executed inside a single ***v*** (i.e. controlled oscillator) period.

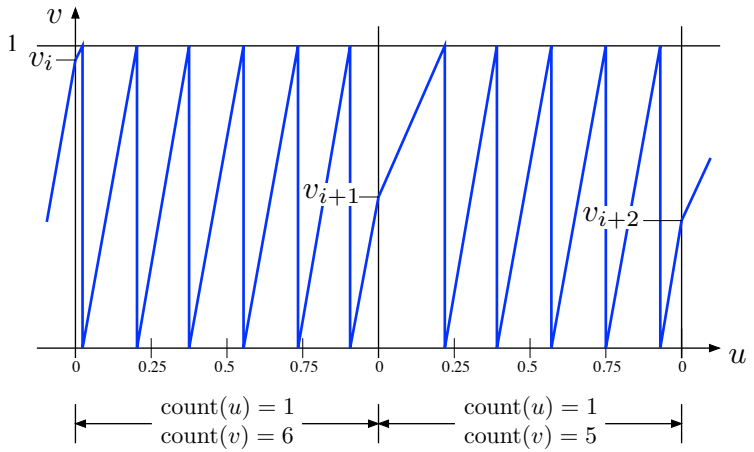


Entropy is extracted from the number of ***u*** (i.e reference oscillator) periods.

Entropy extraction: fast case ($slope_{slow} \geq 1$)

The controlled oscillator ***v*** is always (i.e. also in slow mode) faster than the reference oscillator ***u***.

System iterations are always executed inside a single ***u*** (i.e. clock) period.

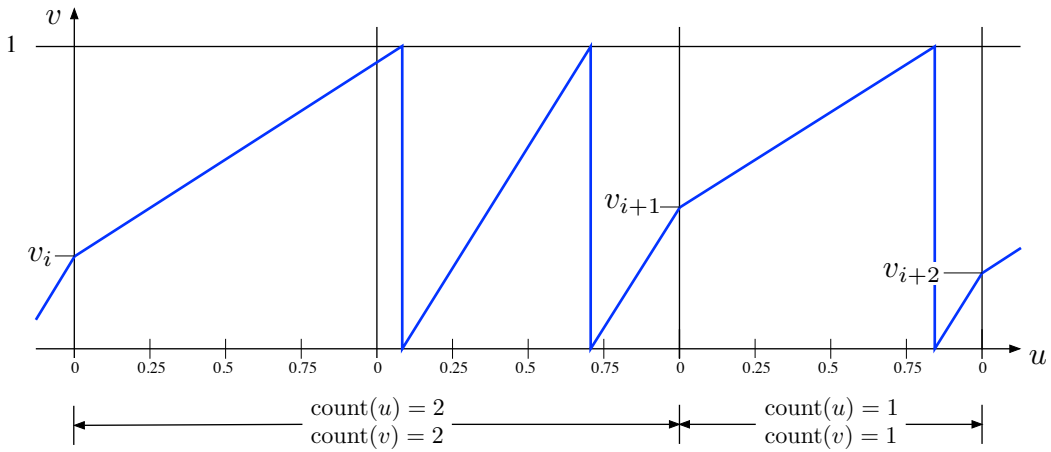


Entropy is extracted from the number of ***v*** (i.e controlled oscillator) periods.

Entropy extraction: intermediate case (slope$_{fast}$ >1 and slope$_{slow}$ < 1)

The controlled oscillator ***v*** can be slower or faster (depending on the slow/fast mode) than the reference oscillator ***u***.

System iterations can include a different number of both reference oscillator ***u*** and controlled oscillator ***v*** periods.



Entropy is extracted from both the number of ***u*** (i.e. reference oscillator) and ***v*** (controlled oscillator) periods.

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - Intuitive operation description
  - **Formal operation description**
- Post-processing and output entropy evaluation

Entropy extraction and generating partitions

The whole entropy rate of the system can be extracted by generating the output sequence $\hat{x}_i = \hat{x}_{i,0}, \hat{x}_{i,1}$ by means of a **generating partition** of the state space:

$$\hat{x}_{i,0} = \left\lfloor slope_{fast} \left(1 - \mathrm{mod}\left(\frac{1-v_i}{slope_{slow}}, 1\right)\right)\right\rfloor$$

$$\hat{x}_{i,1} = \left\lfloor \frac{1-v_i}{slope_{slow}} \right\rfloor$$

Notice: $\hat{x}_i$ takes a different value for each segment of the map

$$v_{i+1} = \mathrm{mod}\left(slope_{fast}\left(1 - \mathrm{mod}\left(\frac{1-v_i}{slope_{slow}}, 1\right)\right), 1\right)$$

In facts, **the generated sequence defines which segment of the map is visited at each iteration**.

Implementation of the generating partitions

Equivalent partitions can be used in order to simplify the implementation. By means of **reversible transformations** we can obtain:

$$\tilde{x}_{i,0} = \mathrm{mod}\left(1 + \left\lfloor slope_{fast}\left(1 - \mathrm{mod}\left(\frac{1-v_i}{slope_{slow}}, 1\right)\right)\right\rfloor, M\right)$$

$$\tilde{x}_{i,1} = \mathrm{mod}\left(1 + \left\lfloor \frac{1-v_i}{slope_{slow}} \right\rfloor, M\right)$$

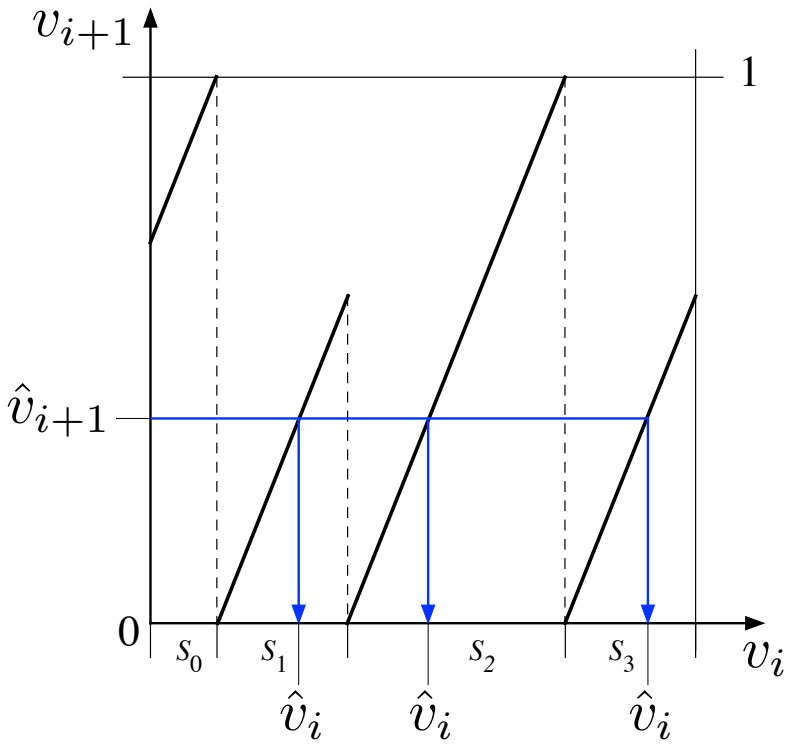
which represent the number of periods per cycle modulo $M = \lceil \log_2 |k| \rceil$ of the controlled and reference oscillator respectively;

$$x_{i,0} = \mathrm{mod}\left(\sum_{j=0}^{i} \tilde{x}_{j,0}, M\right)$$

$$x_{i,1} = \mathrm{mod}\left(\sum_{j=0}^{i} \tilde{x}_{j,1}, M\right)$$

which represent the same quantities but without resetting the counters.

Why all the system entropy is extracted?

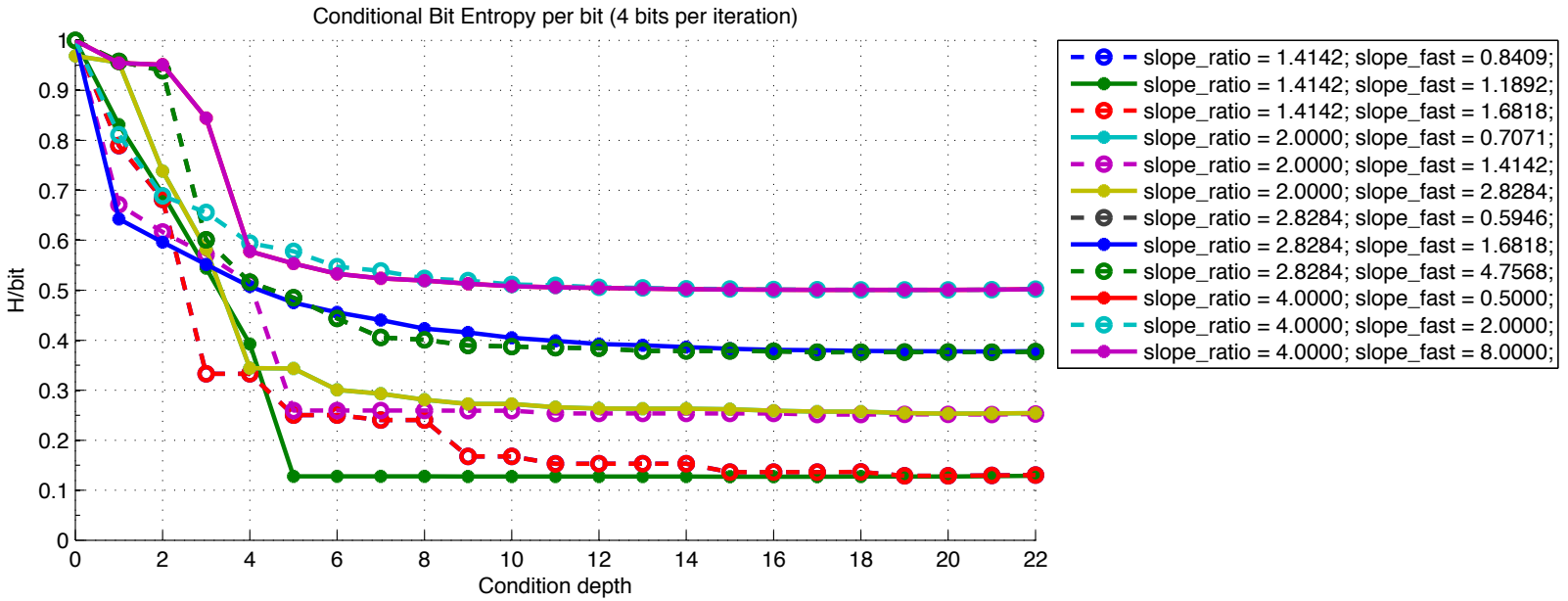


The symbols generated by means of a generating partition allows to **reverse (rewind) the system evolution starting from the current state**.

In our case, a suitable generating partition consists of the partition of the state space in the segments S_0, S_1, S_2, S_3, **where the uni-dimensional map is invertible**.

Since the generated sequence allows to reverse v_i back to v_{i-n} and, since in a reversible transformation entropy si preserved, the $x_{i-n}, \ldots x_i$ sequence must contain the entropy difference between v_{i-n} and v_i .

Source simulation results (slopeRatio = $2^{[0.5, 1.0, 1.5, 2.0]}$)



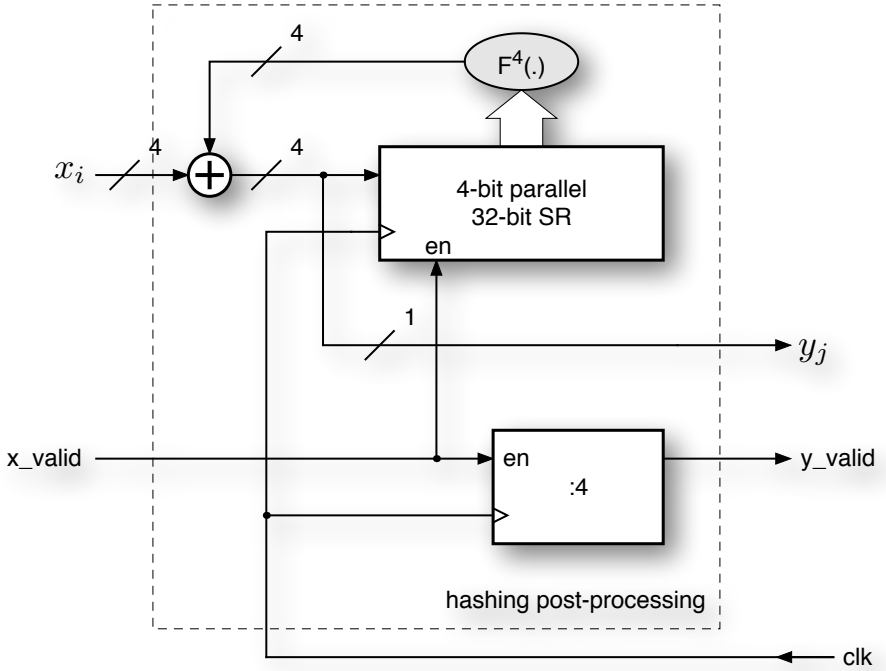
extractor: u, v count with reset;
n_sigma = 1.00e-06;
iterations = 4.00e+08;

Expected entropy per bit: $\log_2(\text{slopeRatio})/4 = \log_2(2^{[0.5, 1.0, 1.5, 2.0]})/4 =$

$= [0.5, 1.0, 1.5, 2.0]/4 = [0.125, 0.250, 0.375, 0.5]$

- Introduction
- Implementation overview
- Chaotic entropy source modelling
  - Summary of known results
  - Bernoulli map and generalised Sawtooth map
  - Why the noise model does not matter
  - Proposed chaotic system
- Entropy extraction
  - Intuitive operation description
  - Formal operation description
- **Post-processing and output entropy evaluation**

Hashing post-processing implementation



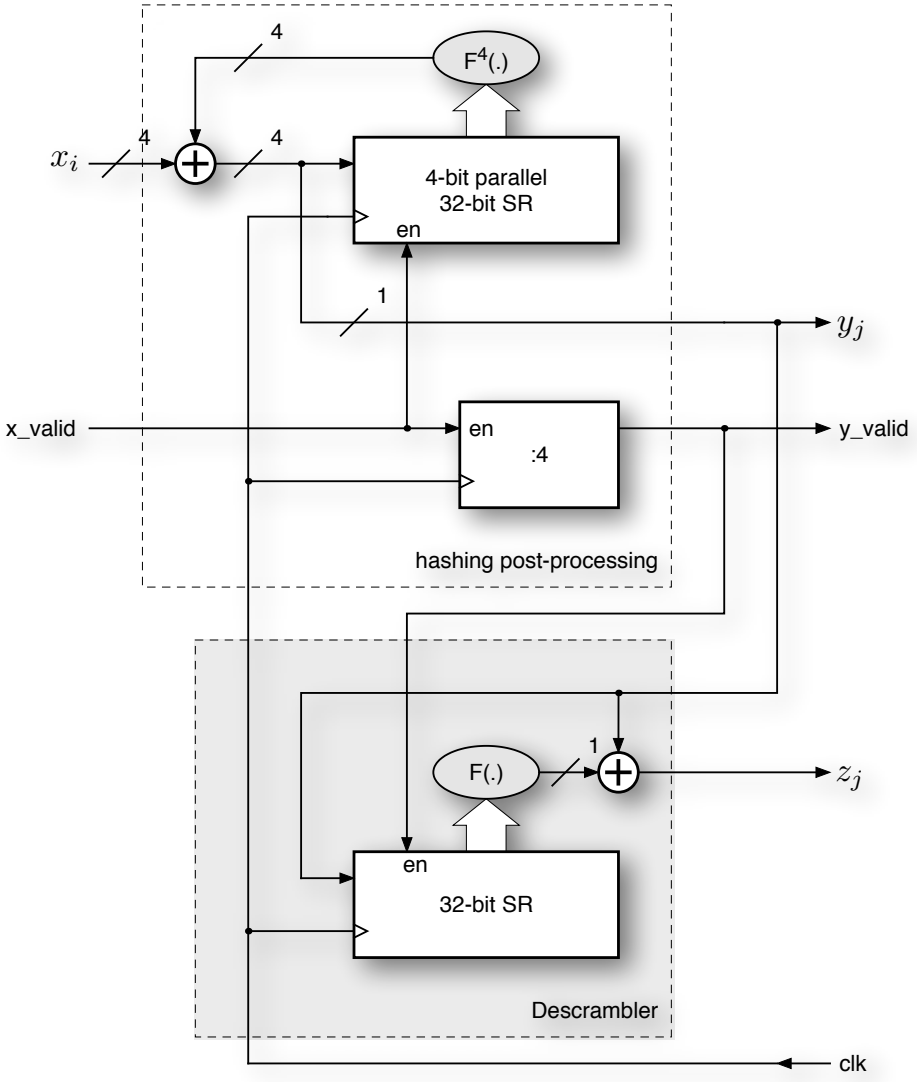
Post-processing consists of a :4 decimated 32-bit LFSR featuring a primitive polynomial feedback.

Implementation is 4-bit parallel in order to accomodate the 4-bit symbols delivered by the entropy source.

Stepping is enabled only when a valid symbol is delivered: **entropy/bit is constant regardless clock frequency**.

Post-processing performs a compression of 1/16 in terms of bits and about 1/5.6 in terms of entropy.

Output entropy estimation by means of a suitable descrambler predictor of post-processing

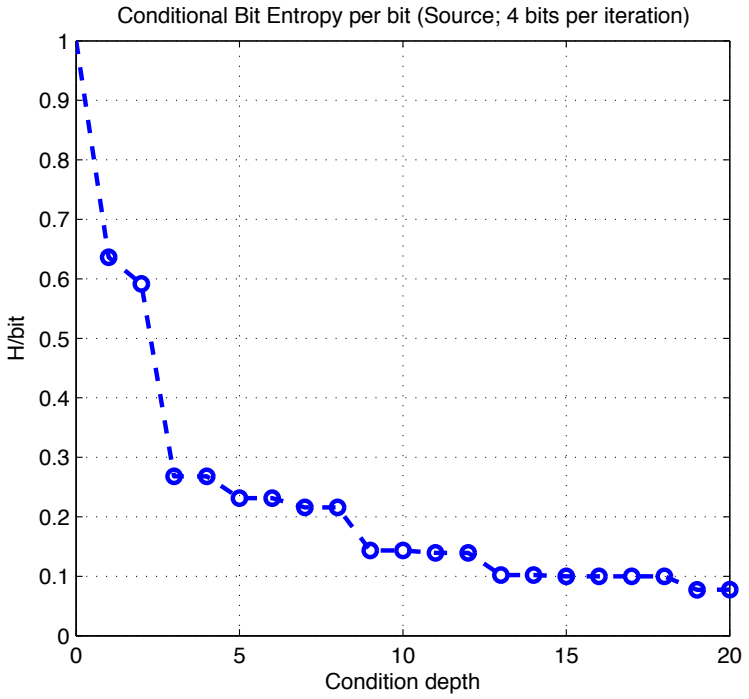


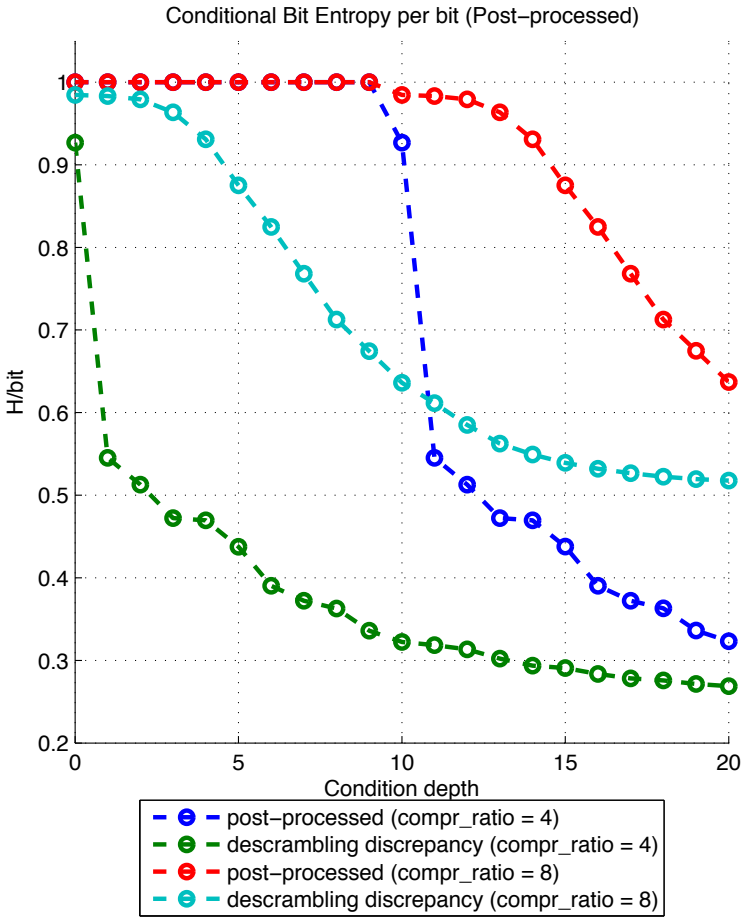
Purpose:
the descrambler performs a reversible transformation of the output sequence Y which **removes the memory introduced by the post-processing LFSR** thus allowing estimating entropy on Z.

Implementation:
the descrambler is a self-synchronising SR featuring the same primitive polynomial as the post-processing LFSR.

Because of a property of m-sequences, despite of the :4 decimation, it can predict the deterministic behaviour of the post-processing LFSR.

Effect of descrambling on post-processing entropy estimation (LFSR_Len = 10; slopeRatio = $2^{0.25}$; slopeFast = $2^{-0.25}$)

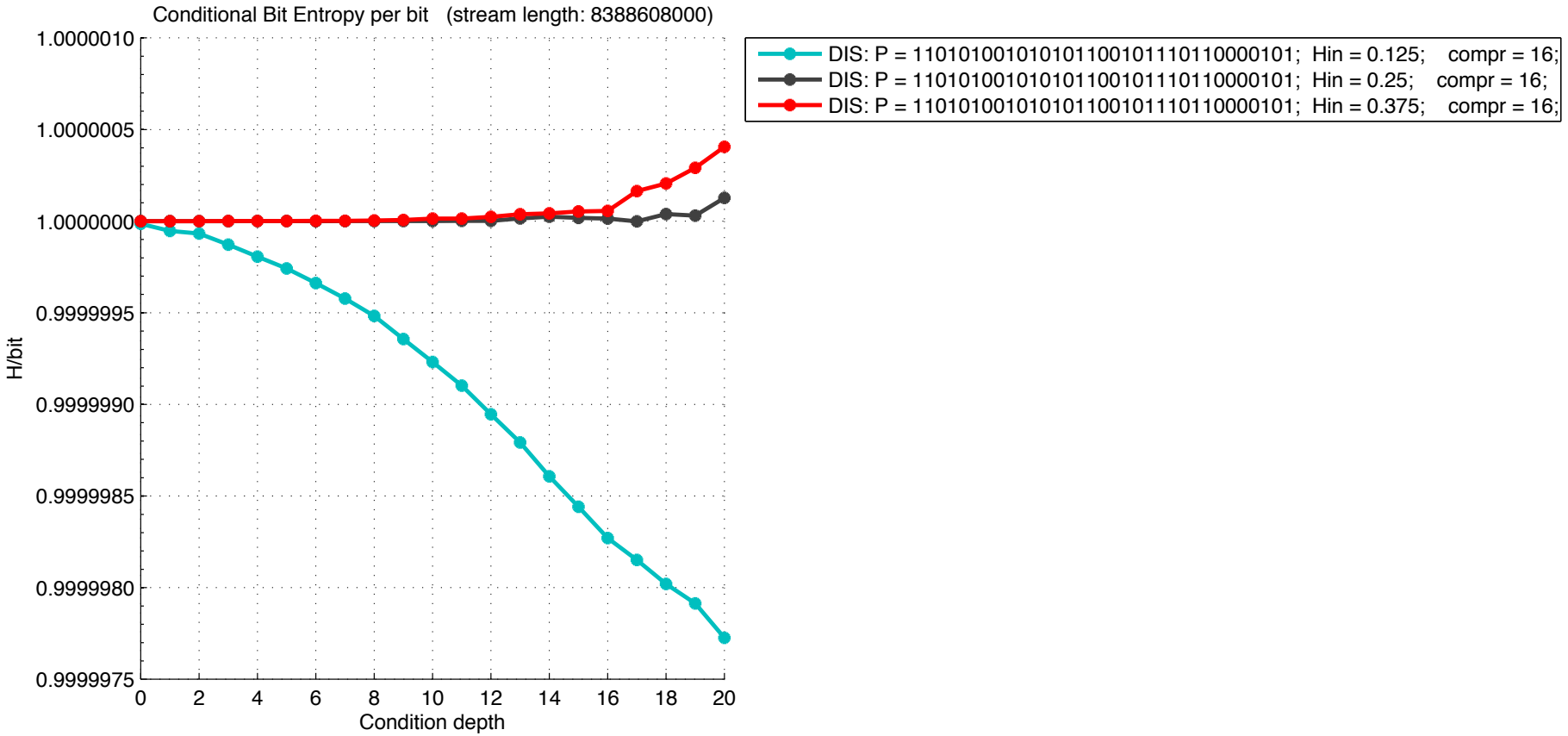




extractor: u, v count with reset;
slope_ratio = 1.189207; slope_fast = 0.840896;
n_sigma = 1.00e-03;
post-proc LFSR: 11011000001; (len = 10);
iterations = 1.00e+09;

Simulation on short LFSR, low entropy and low compression makes evident the effect of descrambling on entropy estimation.

Entropy estimation on descrambled sequences (LFSR_Len = 32; Compr = 16; Hin = [0.125, 0.25, 0.375])



Descrambling allows to detect a lack of entropy (cyan curve) despite the length of the post-processing LFSR is 32.

Conclusions

- Stable and robust entropy generation
- Full digital implementation
  - no additional vulnerability with respect of a P-RBG
  - indistinguishable in the sea-of-gates
- Much more efficient than a P-RNG
- Straightforward entropy estimation (both on source and output data)

Simplicity is a solved complexity

Constantin Brâncuși

Romanian sculptor 1876 – 1957

SW used for source simulation, post-processing and entropy estimation is available under request.