Chaotic Entropy Sources for a New Generation of Random Bit Generators

Marco Bucci marco.bucci@infineon.com
European Cyber Week 2024
Rennes-France

Outline

- **Introduction**
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - Invariant distribution, not a matter of noise
  - A didactical example
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - Chaotic oscillator
  - Entropy extraction and results
- Conclusions

Randomness is a matter of physics, rather than cryptography: an incorrect starting point leads to confusion and poor results

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.“

John von Neumann

However, because this problem has been mainly addressed in the computer science and cryptographic community, it is being tackled by focusing on cryptography (i.e. deterministic transformations), rather than on **physics** where it is a **fundamental and well-investigated problem**

This improper approach to the problem is already evident from the definitions:

Pseudo-random (deterministic) generators are (improperly) called “Random”
Random (non-deterministic) generators are called “true random” to avoid confusion

or, in the attempt to sell as random something that is pseudo-random, “strange” definitions as:

Deterministic Random generator (i.e. pseudo-random)
Non-deterministic Random (i.e. random)

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 Techniques	
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?

Chaos theory offers important and well-established results with regard to unpredictable systems

Entropy rate

- A chaotic system has an “intrinsic” entropy rate
- The intrinsic entropy rate is equal to the **Lyapunov Exponent** (Yakov B. Pesin) (i.e. **the entropy rate does not depend on any noise model**)

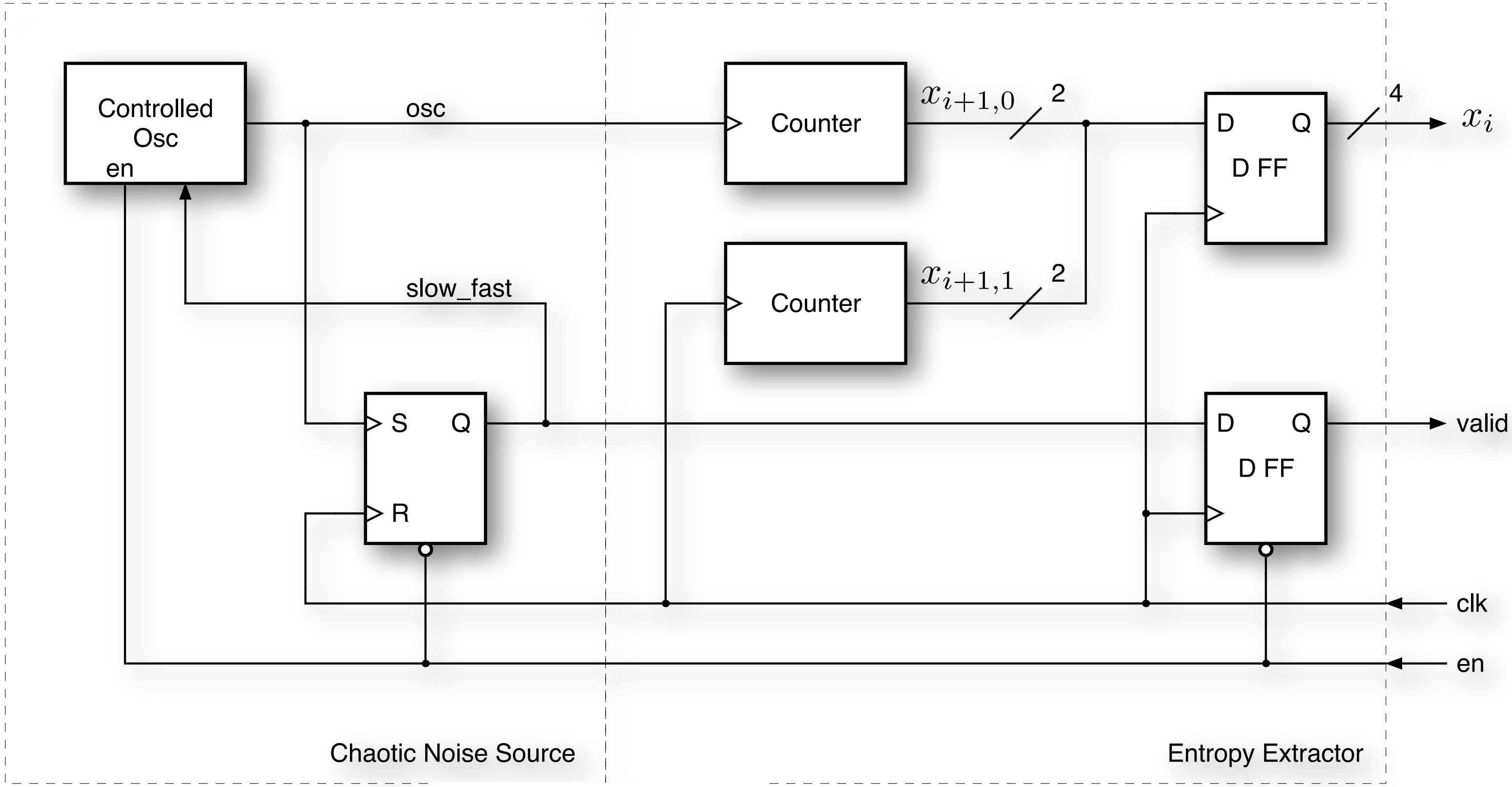
Entropy extraction

- 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**

An implementation example
Two-Speed Oscillator Chaotic Entropy Source



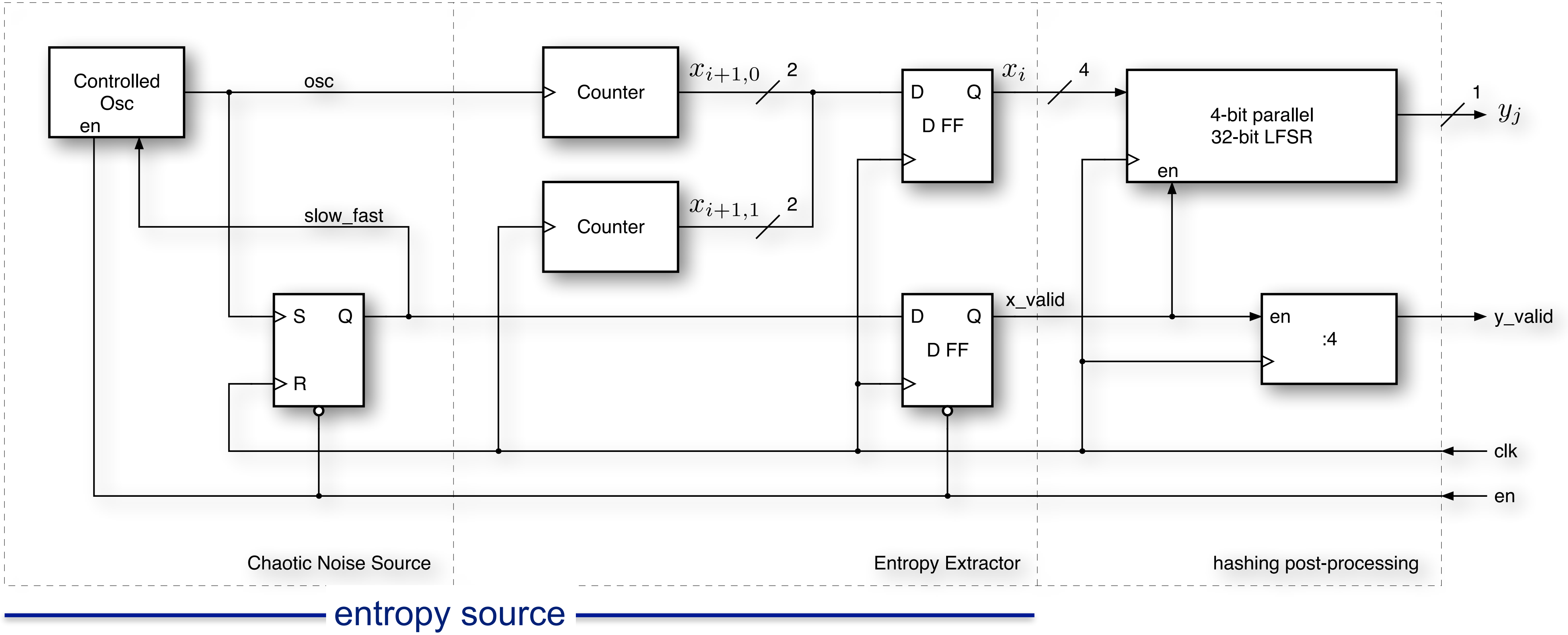
entropy source

- **Small size**:
  comparable with 12 D FF's
- **Low power**:
  some tens of µA
- **High speed**:
  about 1.5 bit entropy per clock

Constant entropy rate:
(**Lyapunov Exponent**)

Full entropy extraction by means of a **Generating Partition**.

An example of Random Bit Generator based on a Chaotic Oscillator Entropy Source



Constant entropy rate: (**Lyapunov Exponent**)

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

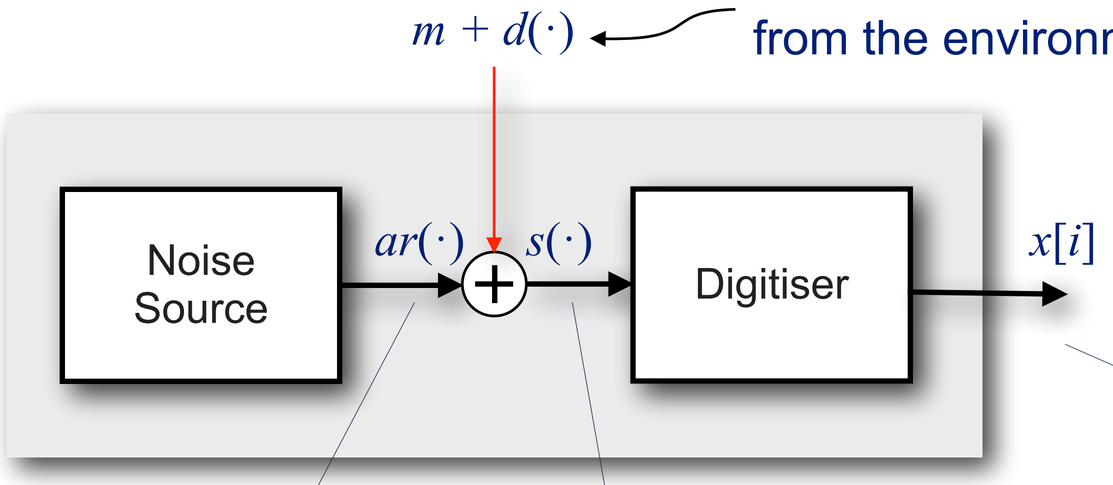
Full entropy output.
Testable by means of a suitable predictor.

Outline

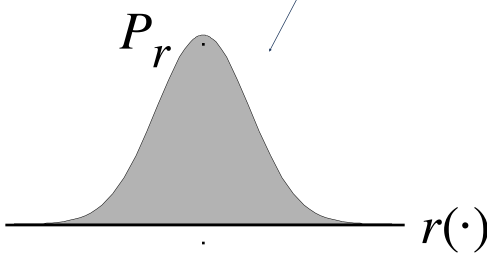
- Introduction
- **Some classic entropy source implementations**
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - Invariant distribution, not a matter of noise
  - A didactical example
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - Chaotic oscillator
  - Entropy extraction and results
- Conclusions

A first, simplistic model of entropy source: ”take some noise and digitise it”

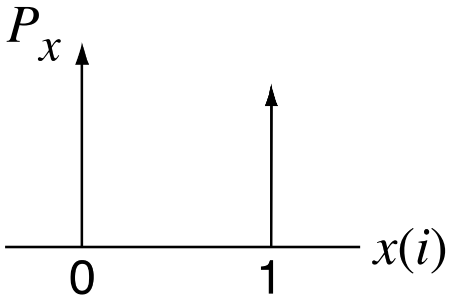
deterministic disturbances
from the environment



$r(\cdot)$: unit (e.g. gaussian) noise
a : noise intensity
m : superimposed offset
$d(\cdot)$: superimposed deterministic disturbance



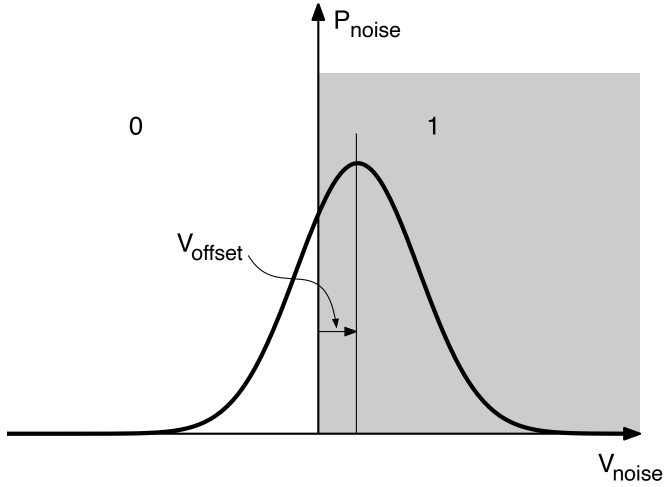
$$s(\cdot) = a\,r(\cdot) + m + d(\cdot)$$

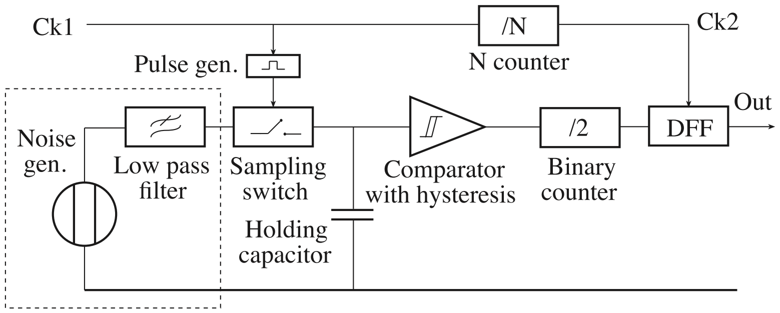


NOTICE: if an attacker can superimpose its own “random” disturbance signal *d*(·), there is no way to detect this attack after digitalisation (no statistical anomalies can be detected)

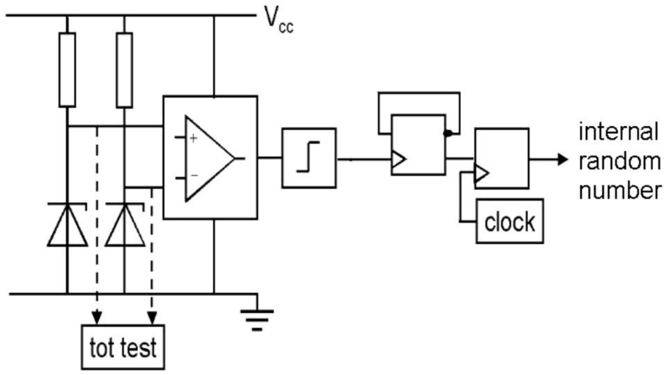
Only on certain conditions, the output entropy and statistic can be estimated (theoretically or empirically) from the (**supposed**) statistic of the Noise Source

Classical implementations:
entropy sources based Direct Noise Sampling

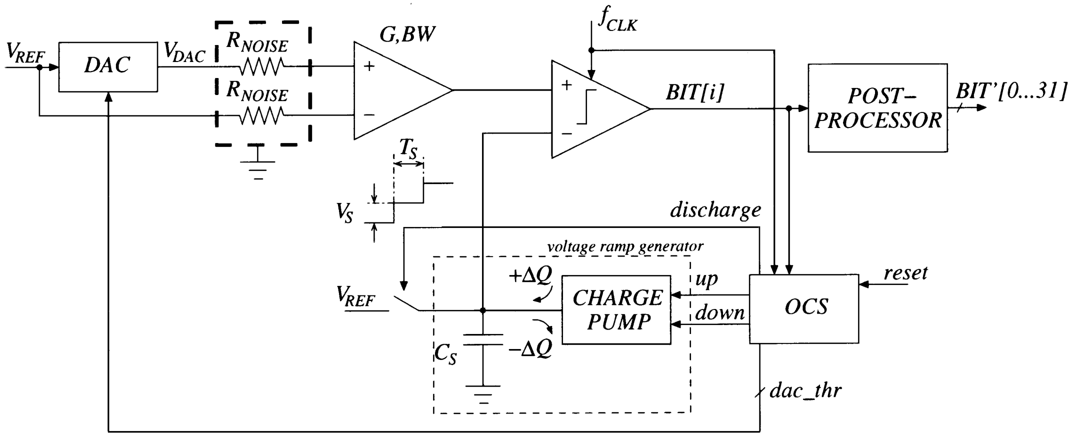




M. Bucci, V. Bagini; CHES 1999



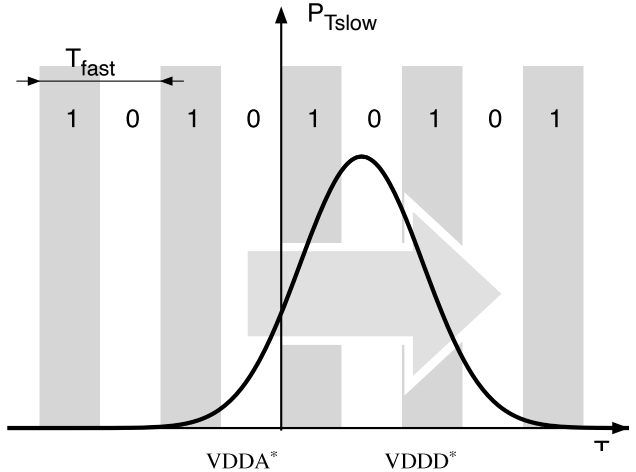
W. Killman, w. Schindler; CHES 2008

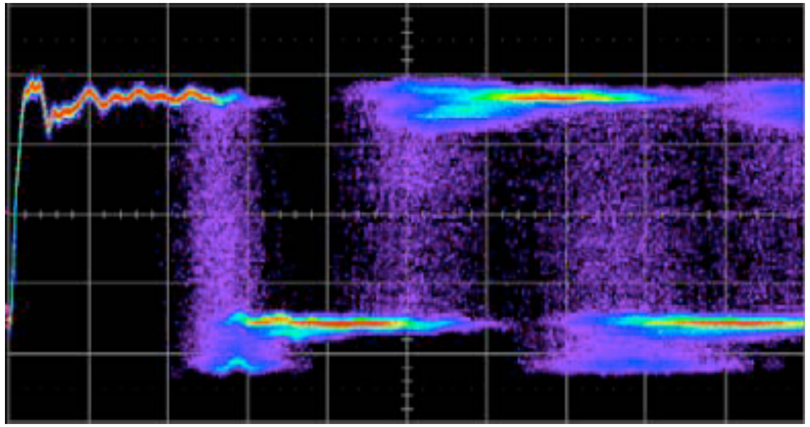


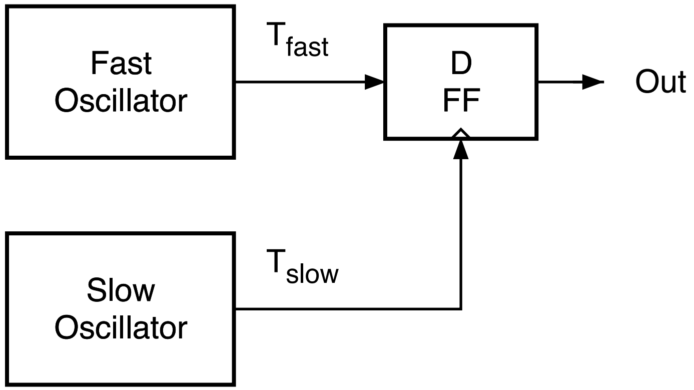
M. Bucci, R. Luzzi et al.; Trans on Circuit and Systems 2003

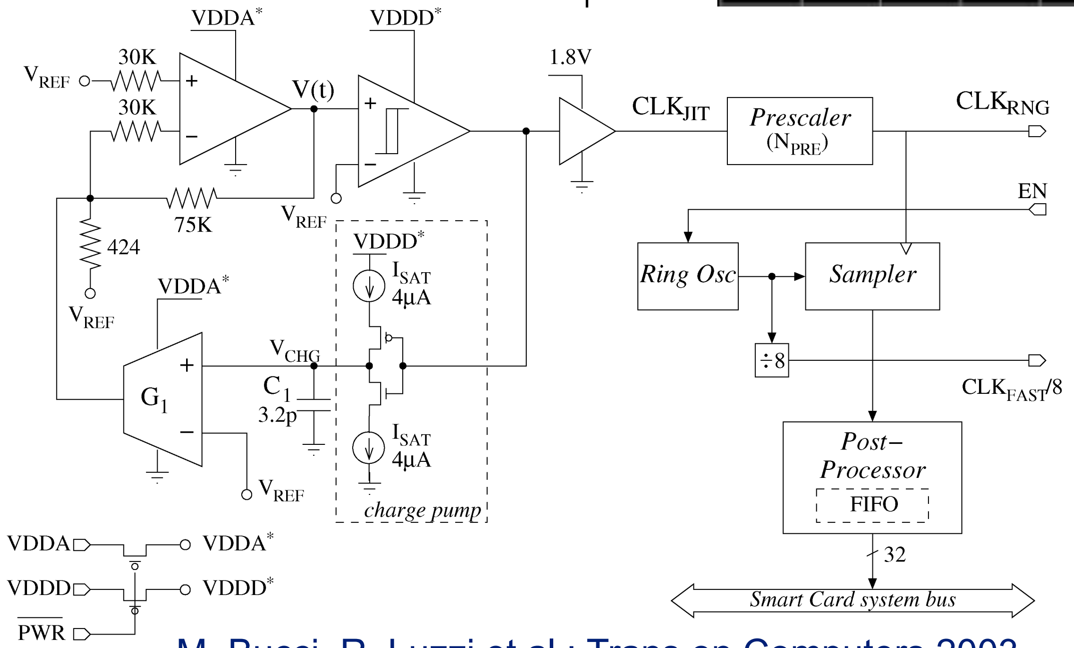
- Easy to evaluate (almost IID)
- Complex:
  - analog design
  - due to small noise, comparator offset must be compensated
- Unsafe: due to small noise and “sign” digitisation, vulnerable to environmental or malicious disturbances
- Slow

Classic implementations:
entropy sources based on analog oscillators (large jitter)





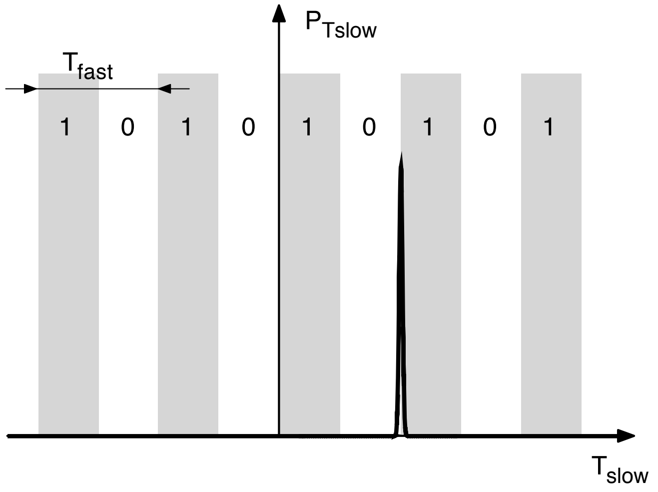


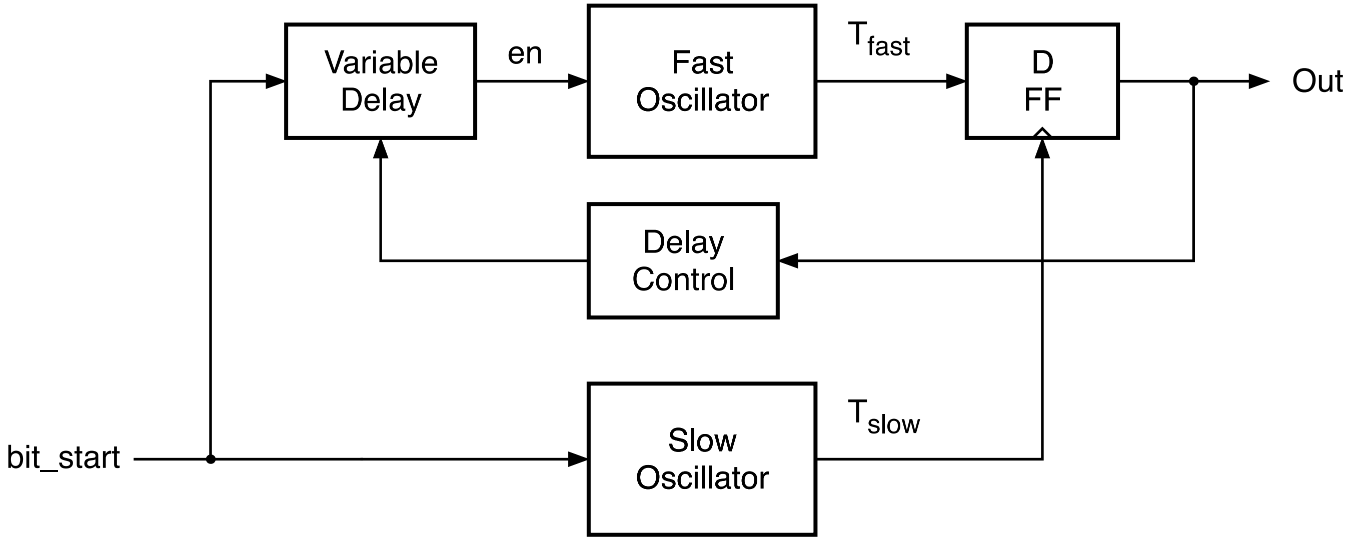


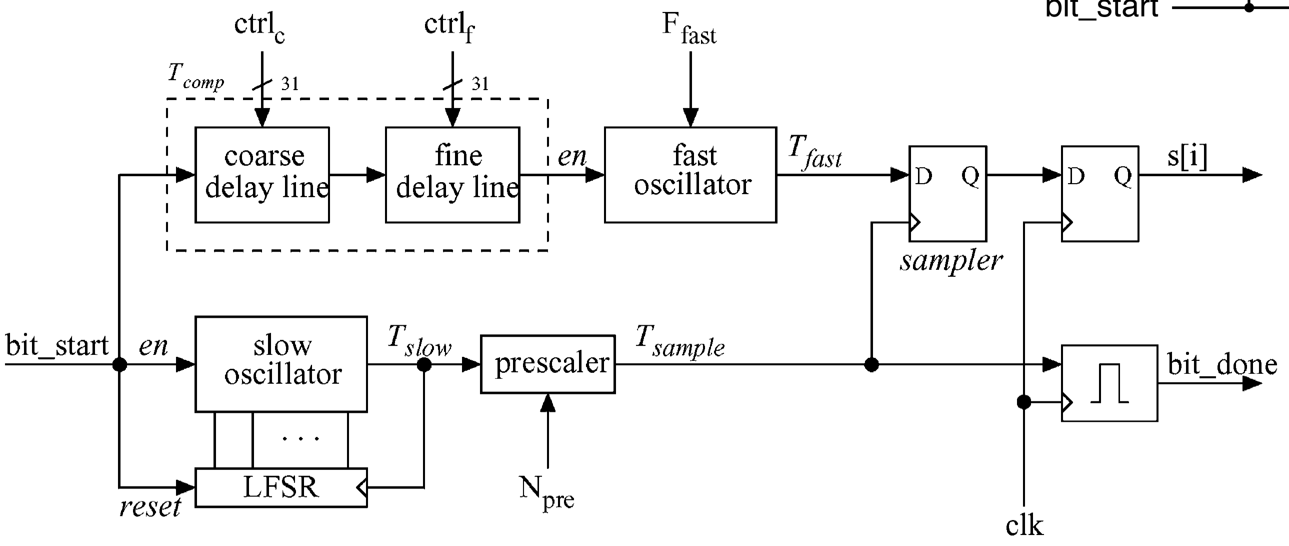
M. Bucci, R. Luzzi et al.; Trans on Computers 2003

- Easy to evaluate (almost IID)
- Robust due to large jitter and 'mod 2' digitisation
- Complex:
  - analog design
- Relatively slow

Classic Implementation: entropy source based on offset compensated ring oscillators (low jitter)



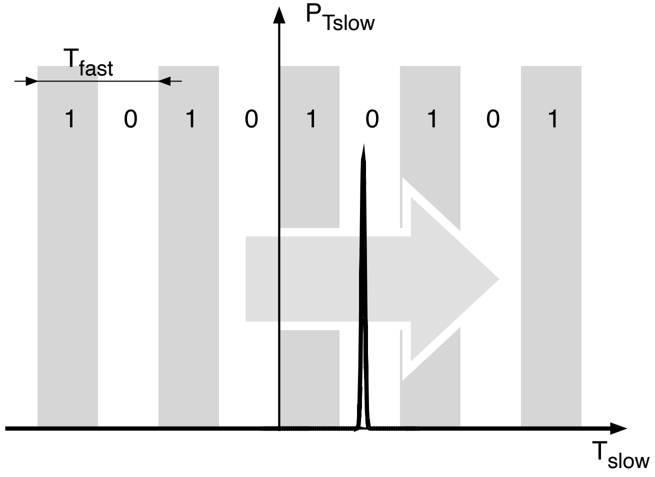




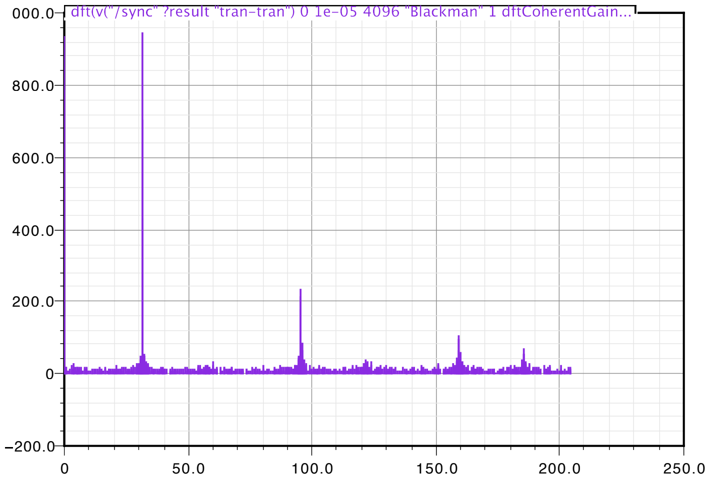
- Easy to evaluate (simple model)
- Quite robust due to 'mod 2' digitisation
- Less complex due to the digital oscillator
- Some complexity due to digital delay lines
- Relatively slow

M. Bucci, R. Luzzi; Trans on Circuit and Systems 2008

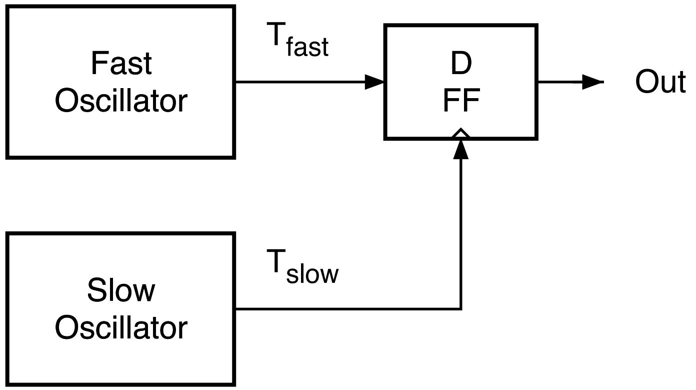
Classic implementations:
entropy sources based on free running ring oscillators (low jitter)



Since the two oscillators are free running, their phases slide towards each other



Frequency spectrum is almost periodic due to the low jitter and the beating between the two frequencies



- Simple implementation
- Output is quasi-periodic
- Difficult to evaluate (long-term dependencies)
- Can be affected by synchronisation with other signals (environmental or malicious)
- Relatively slow

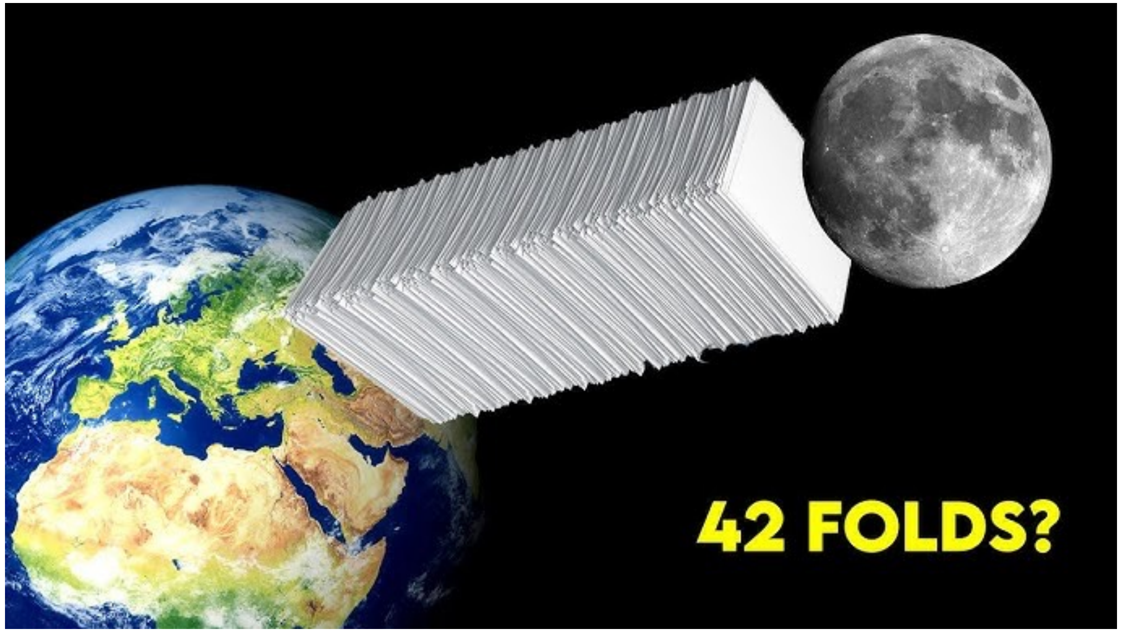
In some implementations, to hide the quasi-periodic behaviour, dozens of oscillators are connected in XOR.

However, this solution is naive and extremely expensive in area and power consumption.

Outline

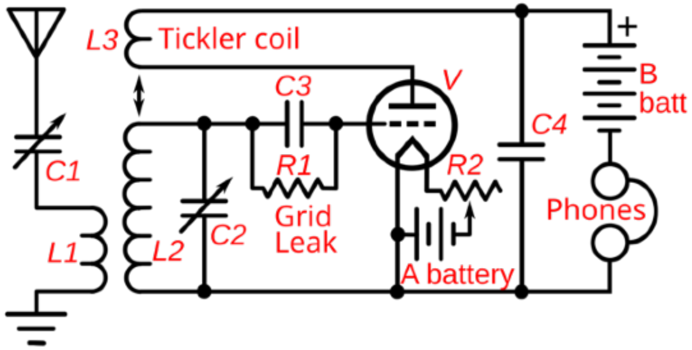
- Introduction
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - **Stretching and folding, the power of exponential growth**
  - Invariant distribution, not a matter of noise
  - A didactical example
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - Chaotic oscillator
  - Entropy extraction and results
- Conclusions

Exponential Growth:
something difficult to imagine even for those who know mathematics

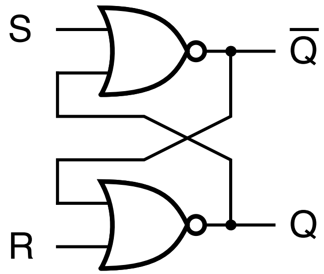


An example of exponential growth: the thickness of a piece of paper folded 42 times reaches the distance between the earth and the moon!

In electronics, it is well known that exponential growth can be achieved simply by means of a positive (regenerative) feedback



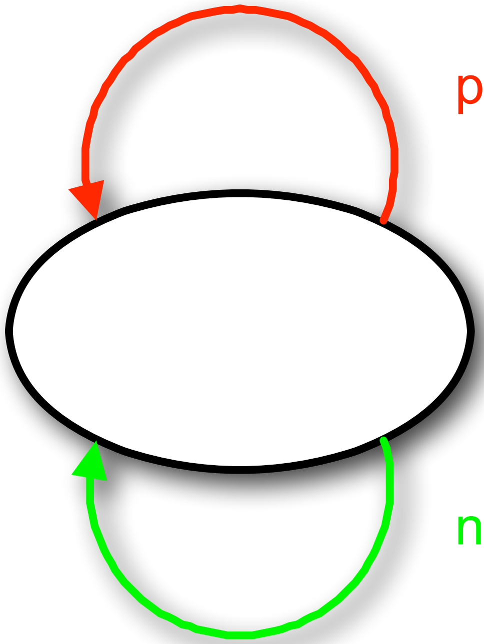
1930's Armstrong regenerative receiver achieved a tremendous sensitivity by exploiting slight positive feedback



Flip Flops achieve a fast (exponential) switching by means of positive reaction

But how to achieve exponential growth without saturation?

Chaotic Systems: exponential growth without saturation



positive feedback
(stretching)

negative feedback
(folding)

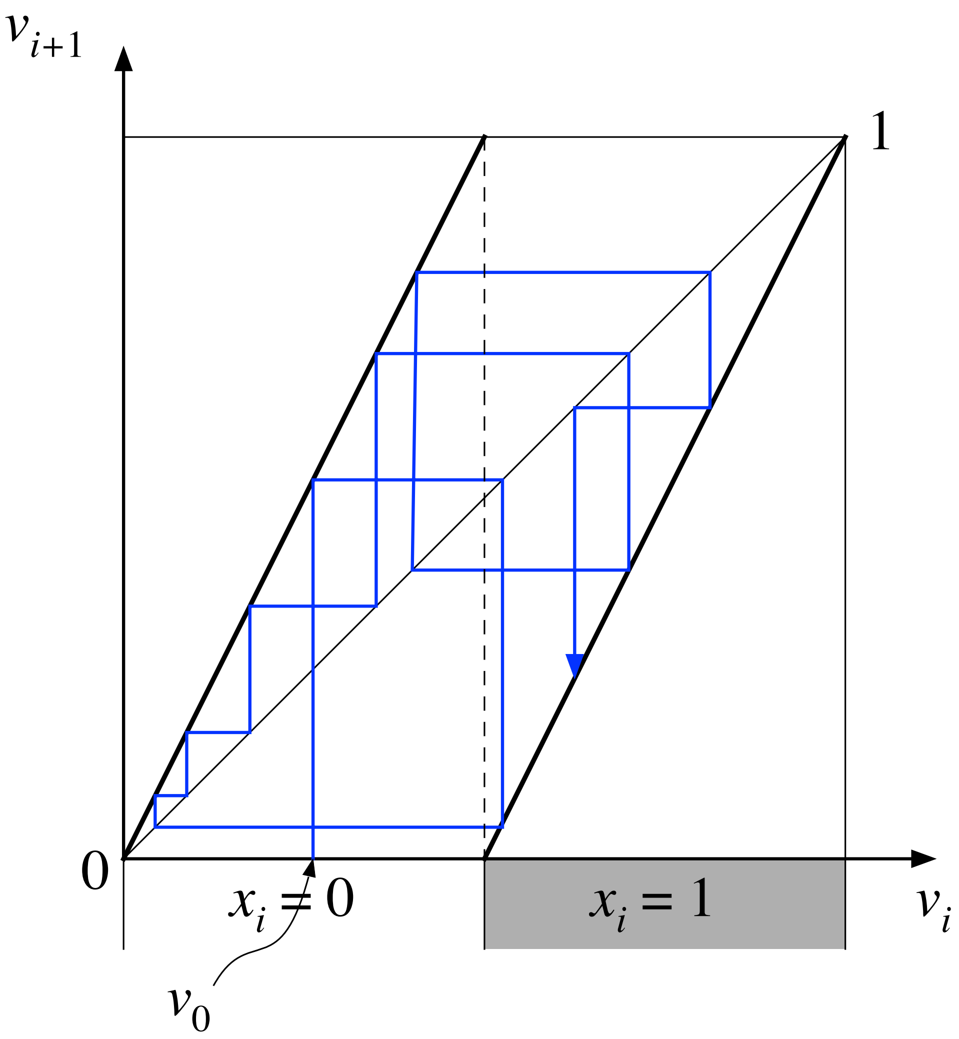
a positive feedback loop exponentially stretches (amplifies) a state variable

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 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:

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

folding stretching

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

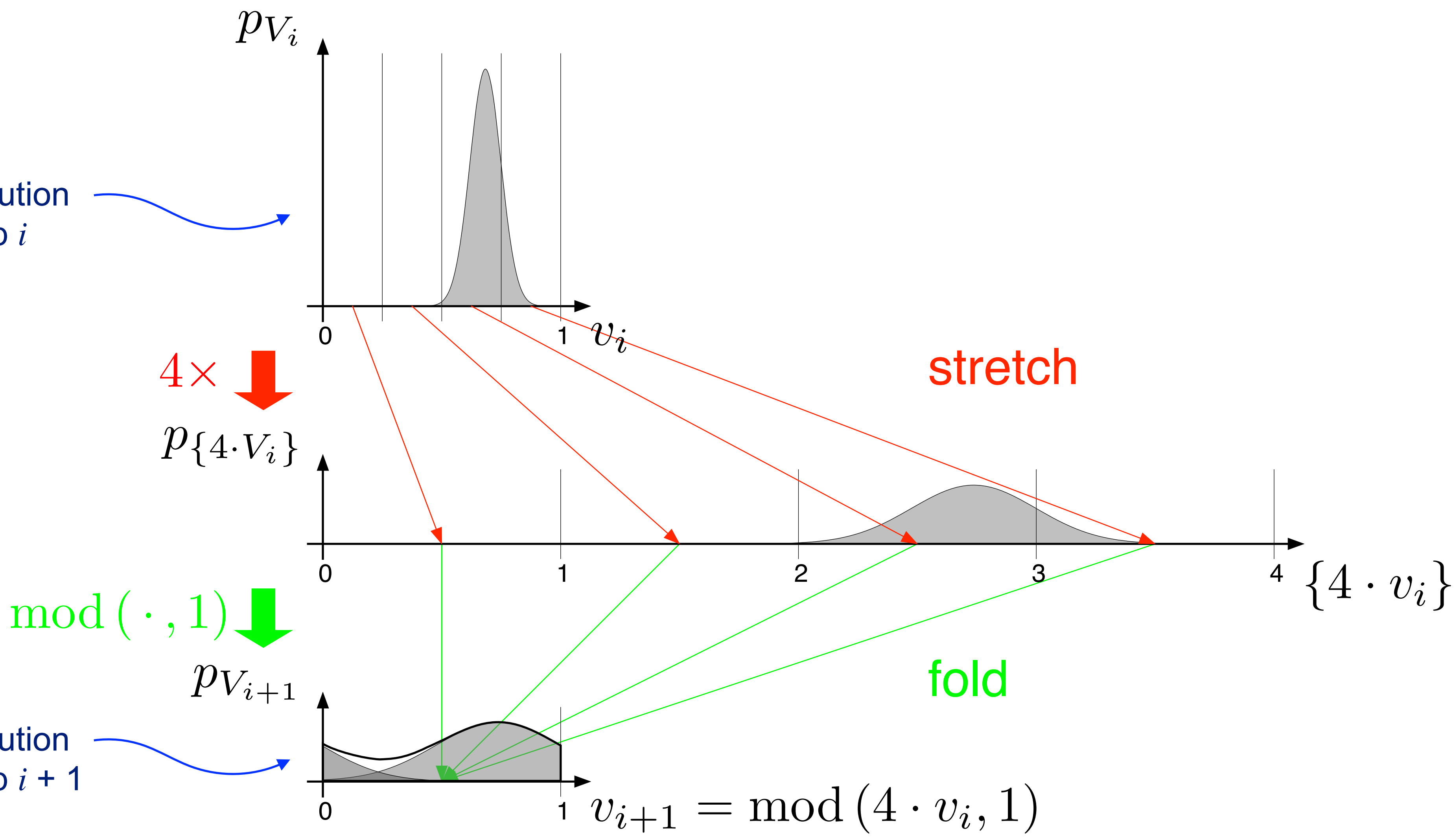
The generated sequence $x_i = \lfloor 2 \cdot v_i \rfloor$ actually consists of the binary representation of the initial state:

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

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

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

State distribution at step i

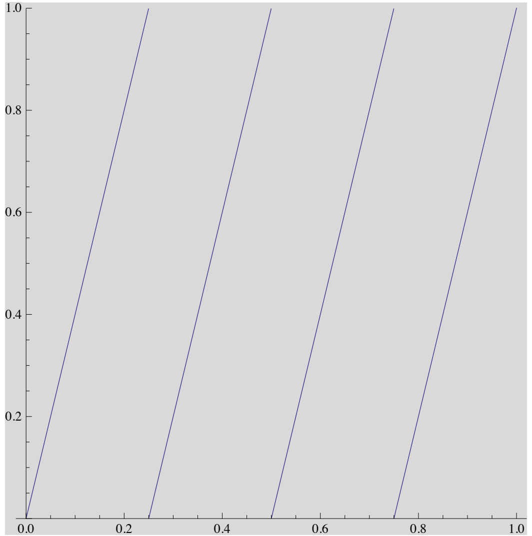


State distribution at step i + 1

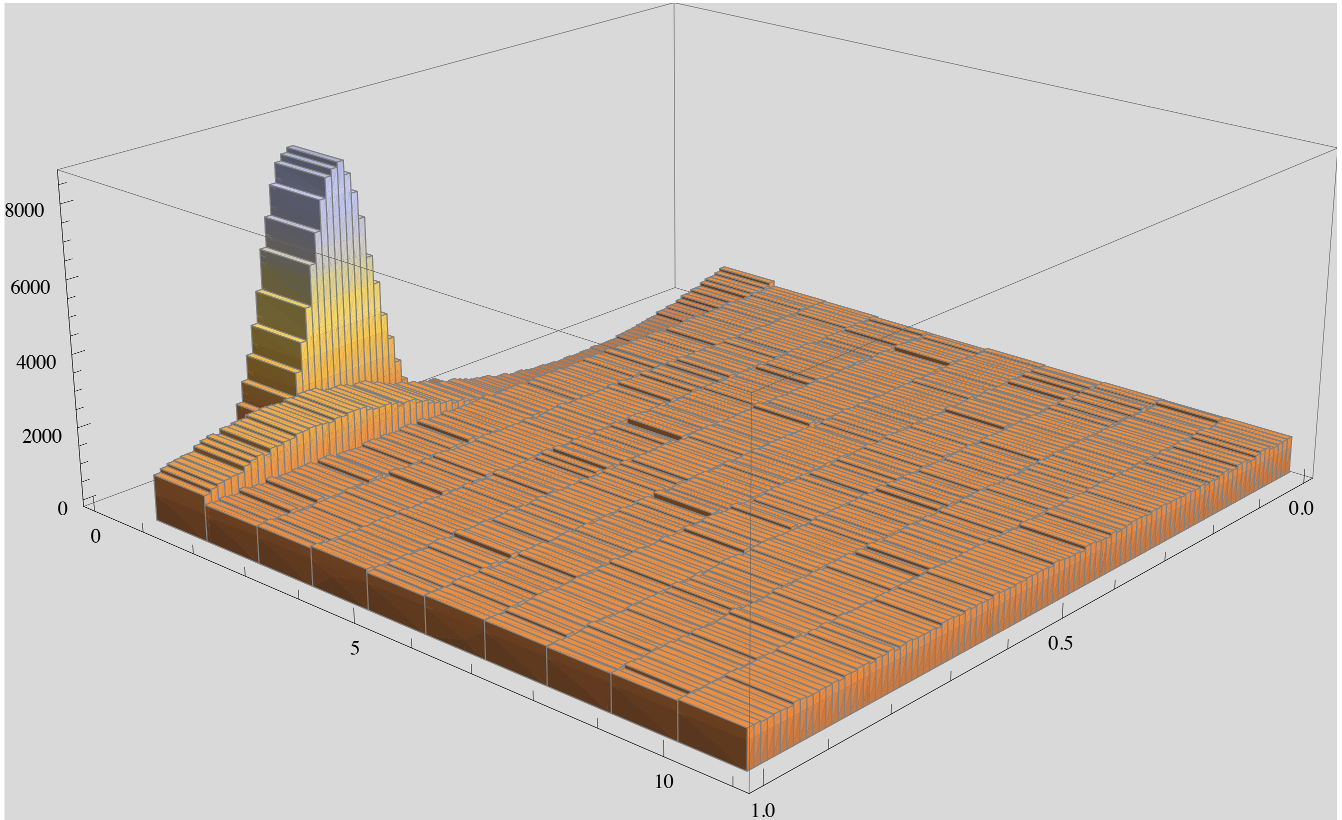
Outline

- Introduction
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - **Invariant distribution, not a matter of noise**
  - A didactical example
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - Chaotic oscillator
  - Entropy extraction and results
- Conclusions

The state converges to an invariant distribution: example $v_{i+1} = \mathrm{mod}(k \cdot v_i\,, 1)$ for $k = 4$



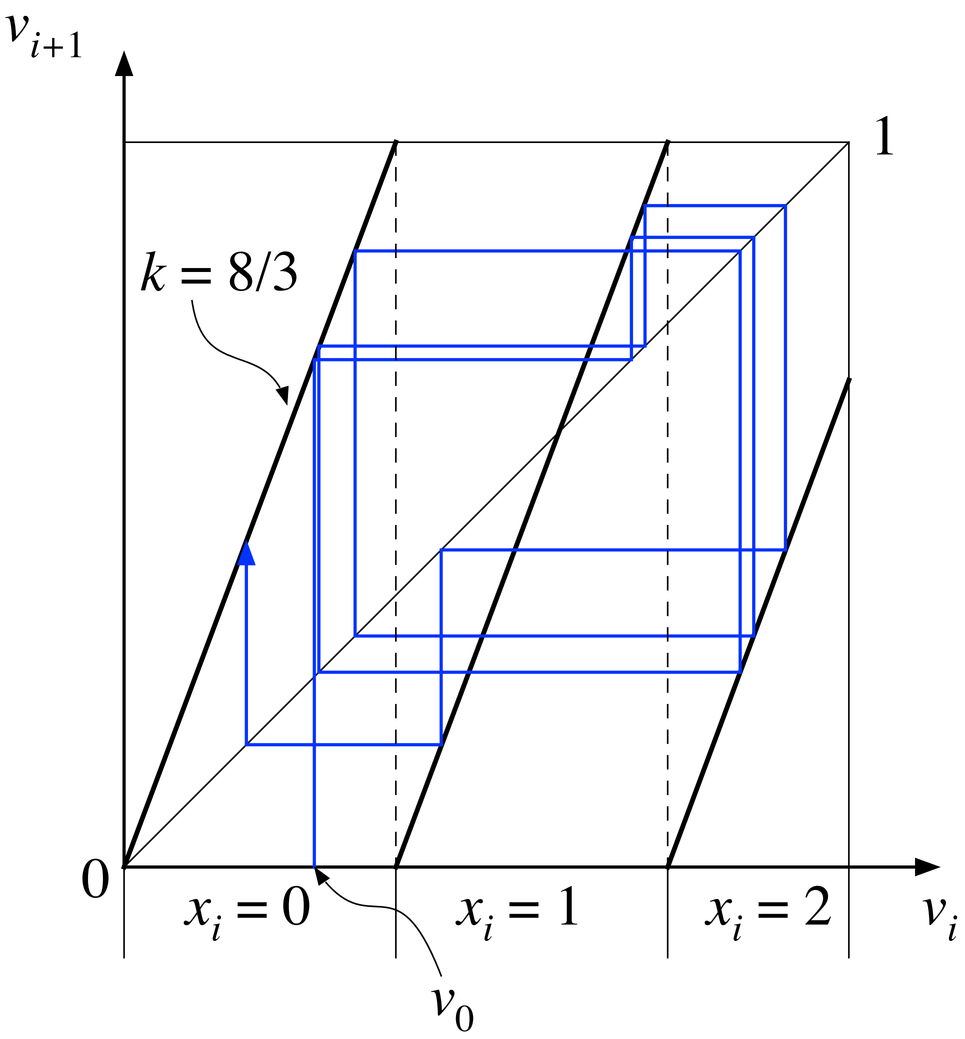
Iteration map:
$v_{i+1} = \mathrm{mod}(4 \cdot v_i\,, 1)$



State distribution vs time step

Whatever the initial state or perturbations during evolution, the system converges exponentially to an **invariant** (i.e. stationary) **distribution** that, since ***k*** is integer (folding completely overlaps), is also **uniform**

Generalised Sawtooth Map



Generalisation of the Bernoulli map:

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

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

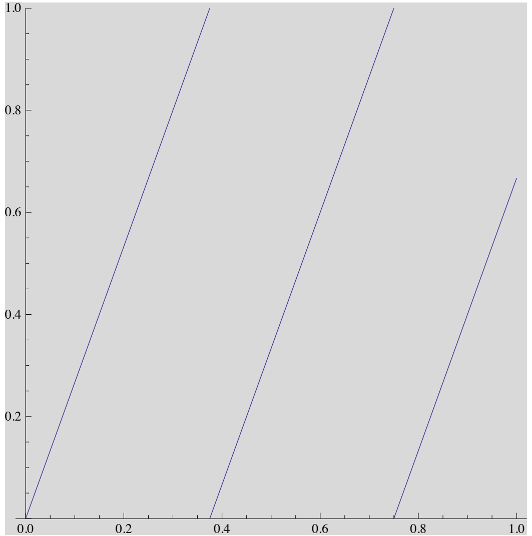
$$|k| > 1$$

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:

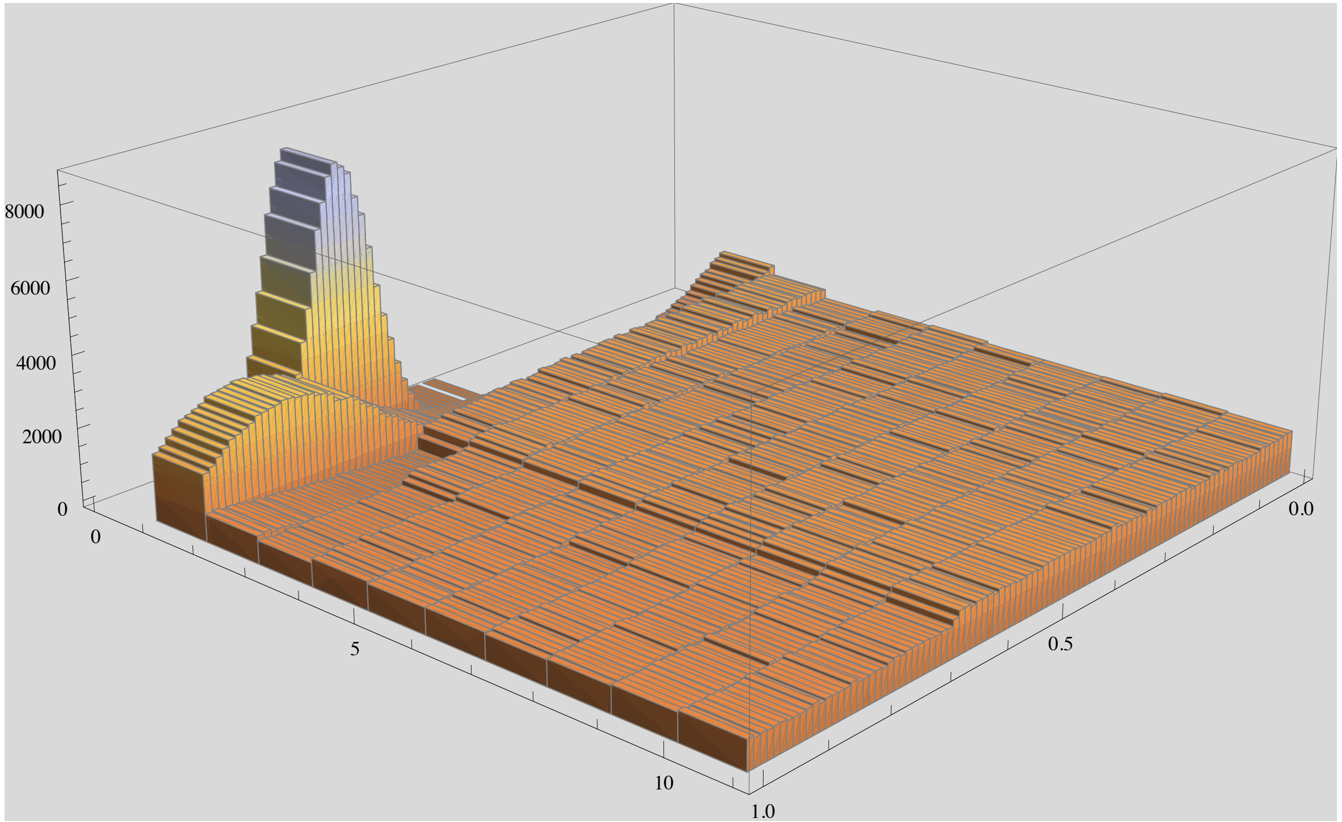
$$h(X) = \log_2 |k|$$

The state converges to an invariant distribution: example $v_{i+1} = \mathrm{mod}(k \cdot v_i\,, 1)$ for $k = 8/3$



Iteration map:

$v_{i+1} = \mathrm{mod}((8/3) \cdot v_i\,, 1)$



State distribution vs time step

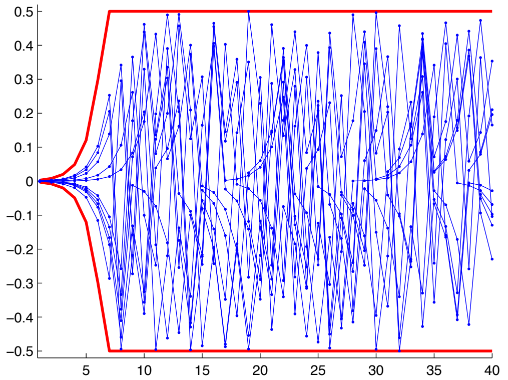
Whatever the initial state or perturbations during evolution, the system converges exponentially to an **invariant** (i.e. stationary) **distribution** that, since $\boldsymbol{k}$ is not integer, is **not uniform**

Outline

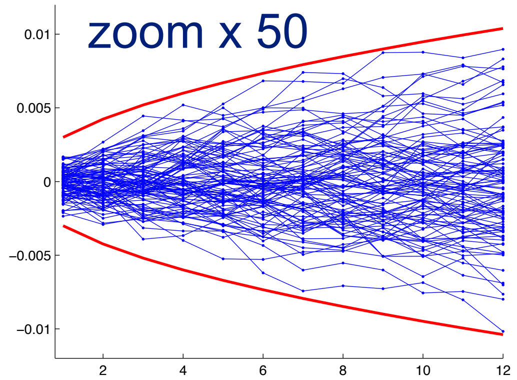
- Introduction
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - Invariant distribution, not a matter of noise
  - **A didactical example**
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - Chaotic oscillator
  - Entropy extraction and results
- Conclusions

Comparing noise effects on **Chaotic**, **Free Running** and **PLL** oscillators: a chaotic oscillator is an “anti PLL”

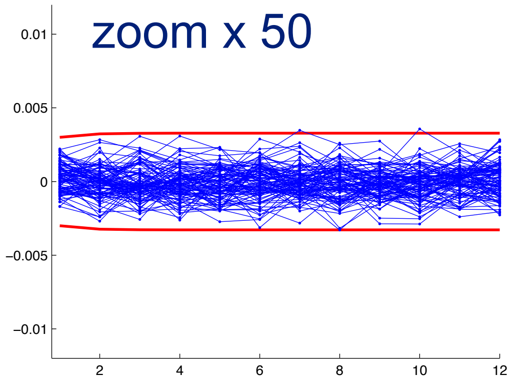
Jitter vs Iteration



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

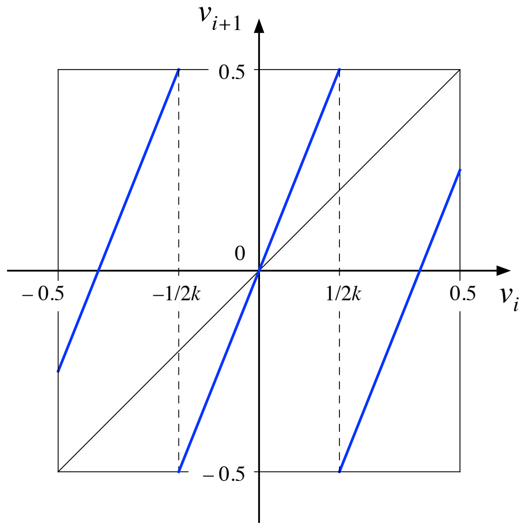


$k = 1 \Rightarrow$ Free Running

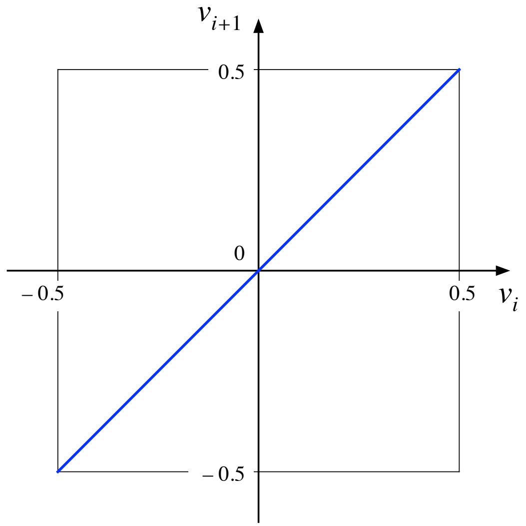


$k = 2^{-1.3} > 1 \Rightarrow$ Phase Locked

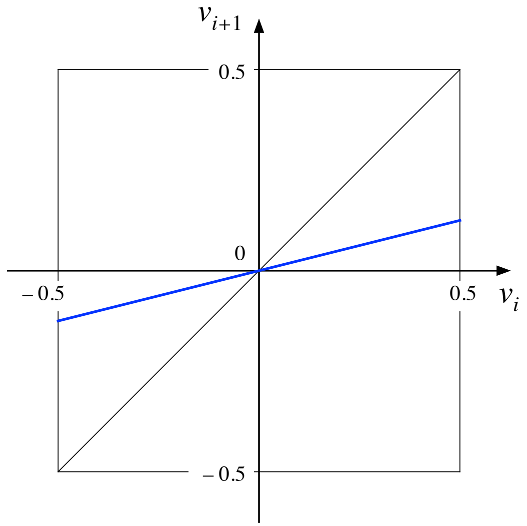
Phase Feedback (iteration Map)



Positive Feedback

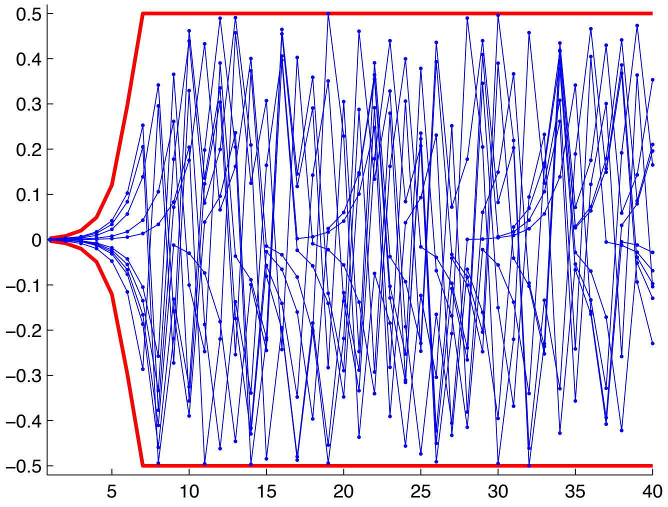


No Feedback

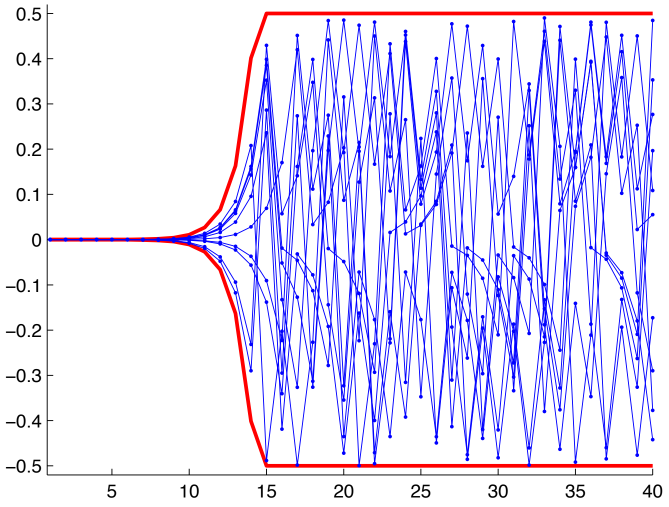


Negative Feedback

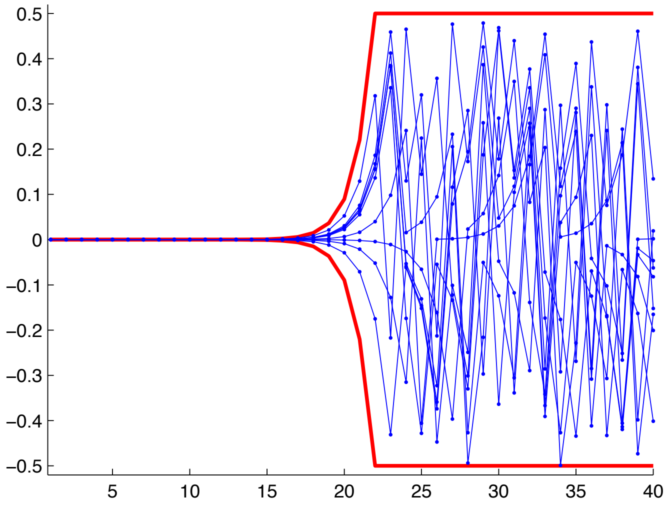
Noise intensity does not practically matter



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



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



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

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 extracted and 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 $H(X) = \log_2 k$.

Outline

- Introduction
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - Invariant distribution, not a matter of noise
  - A didactical example
  - **Uncertainty expansion and entropy rate, also not a matter of noise**
- An implementation example
  - Chaotic oscillator
  - Entropy extraction and results
- Conclusions

How entropy is produced at each step?
Why the entropy rate coincides with the Lyapunov Exponent?

The entropy rate of the system consists of the amount of information obtained with each observation, given the prediction due to the previous observation:

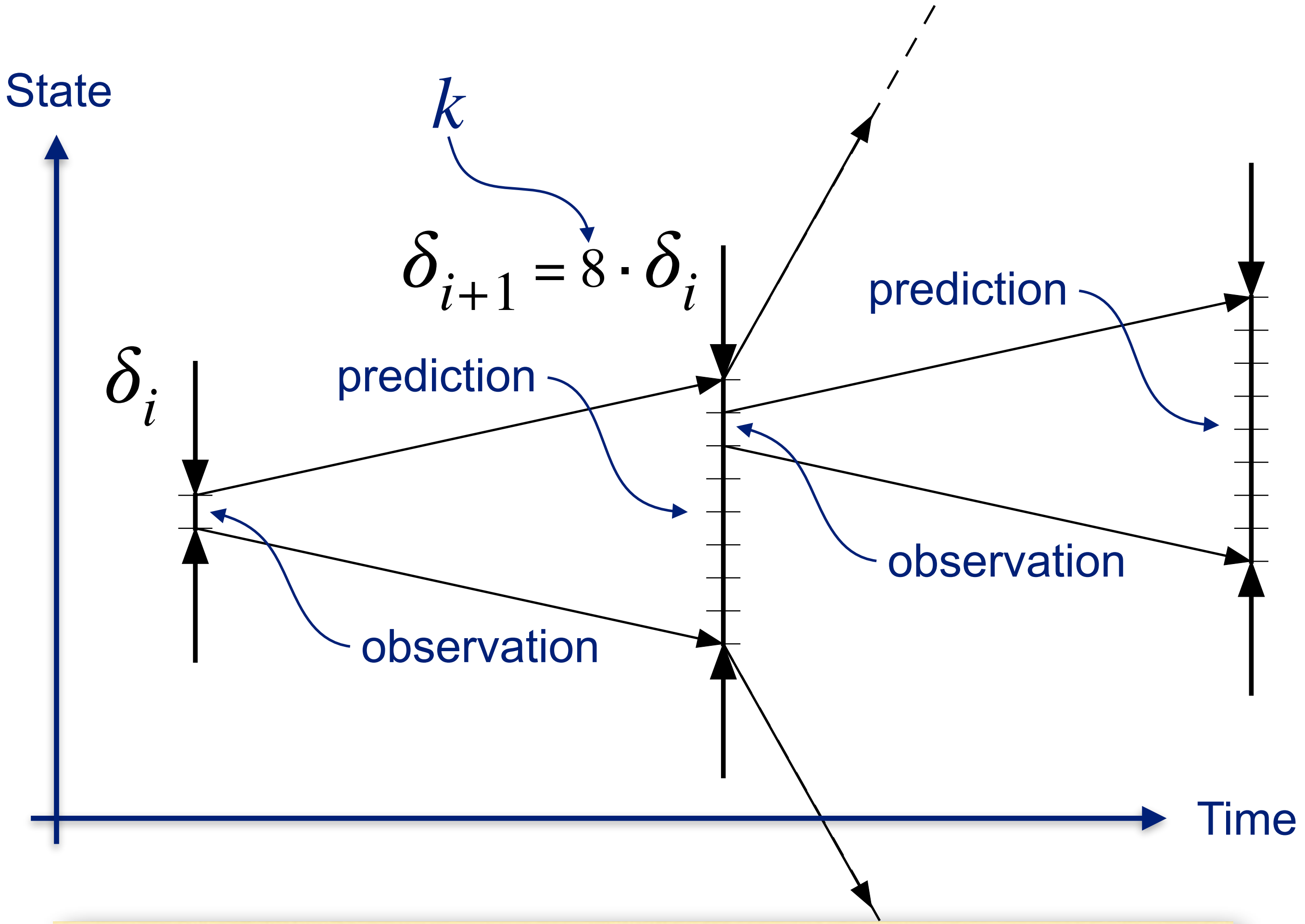
$$H = \log \frac{Prediction\_error}{Observation\_error}$$

If the error (uncertainty) expands at a rate k

$$H = \log_2 k$$

The **Lyapunov Exponent** λ is defined as:

$$\lambda = \ln k$$

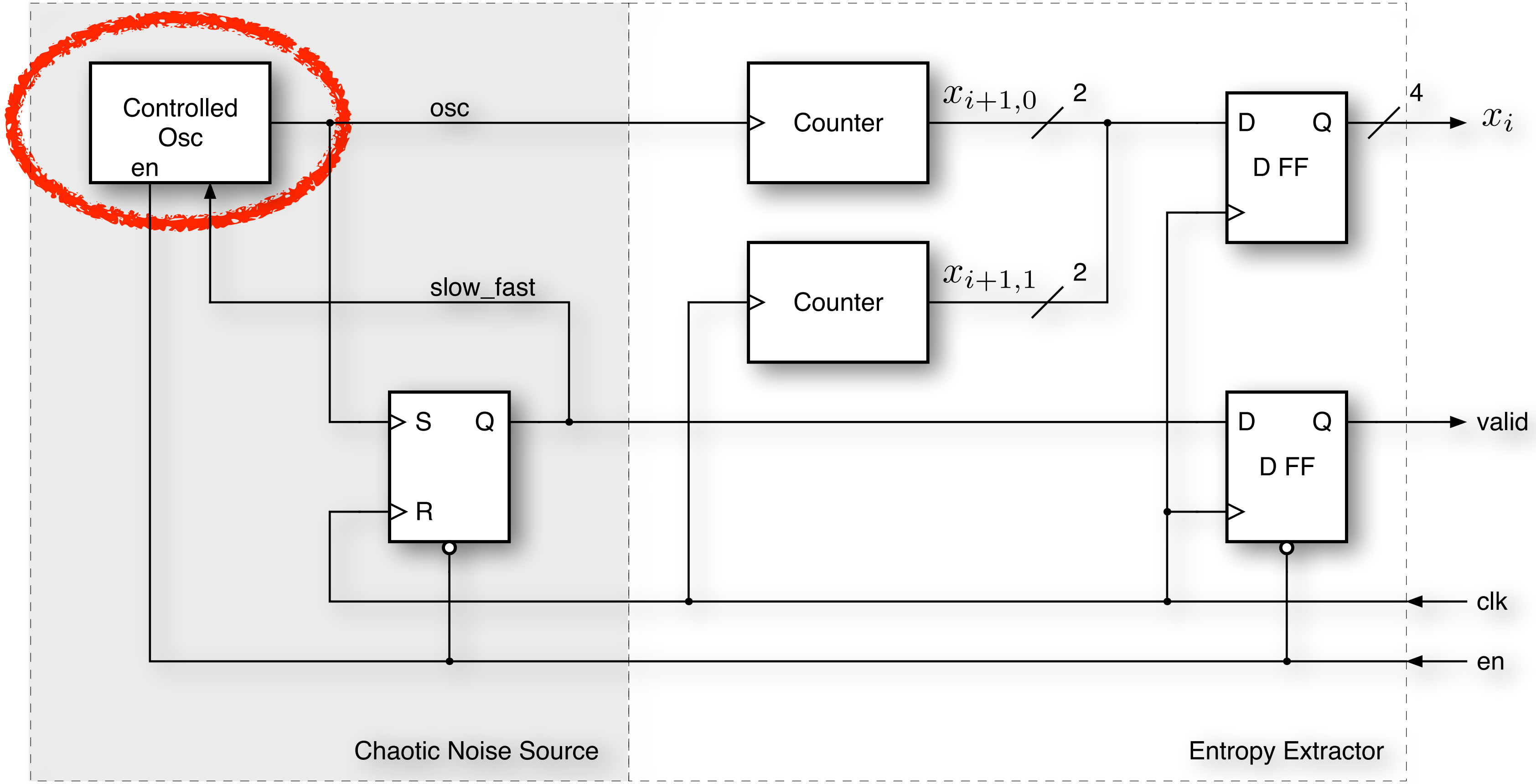


In this example as uncertainty expands at a rate $k = 8$, each observation produces $\log_2 8 = 3$ bits of entropy

Outline

- **Introduction**
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - Invariant distribution, not a matter of noise
  - A didactical example
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - **Chaotic oscillator**
  - Entropy extraction and results
- Conclusions

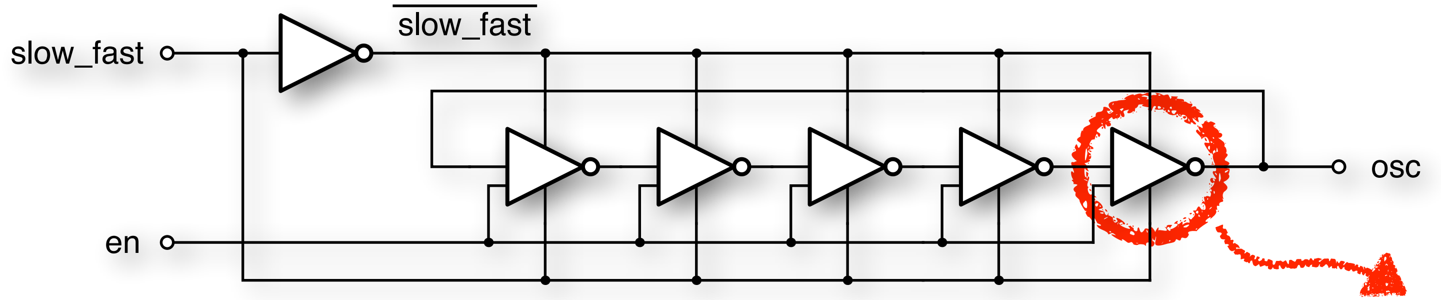
Noise source:
Chaotic oscillator based on a two-speeds controlled oscillator



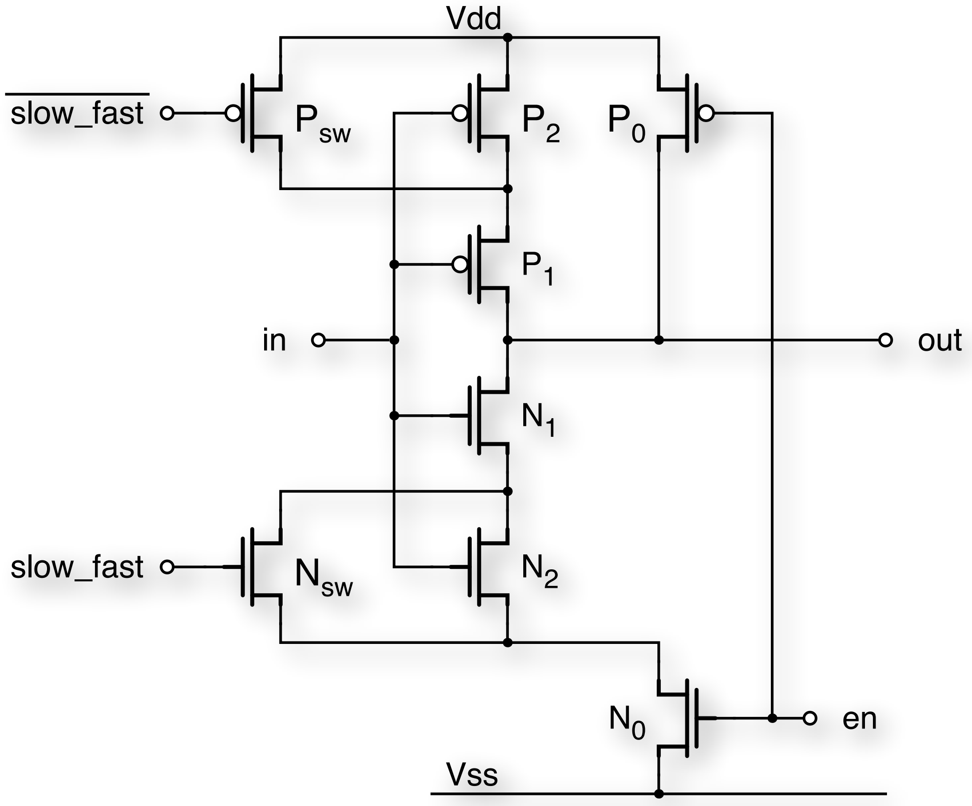
Chaotic Oscillator

The whole system depends on a **single and not critical parameter**: the ratio between fast and slow frequency of the controlled oscillator

Implementation of the two-speeds controlled oscillator: ring oscillator version



Two-speed ring oscillator



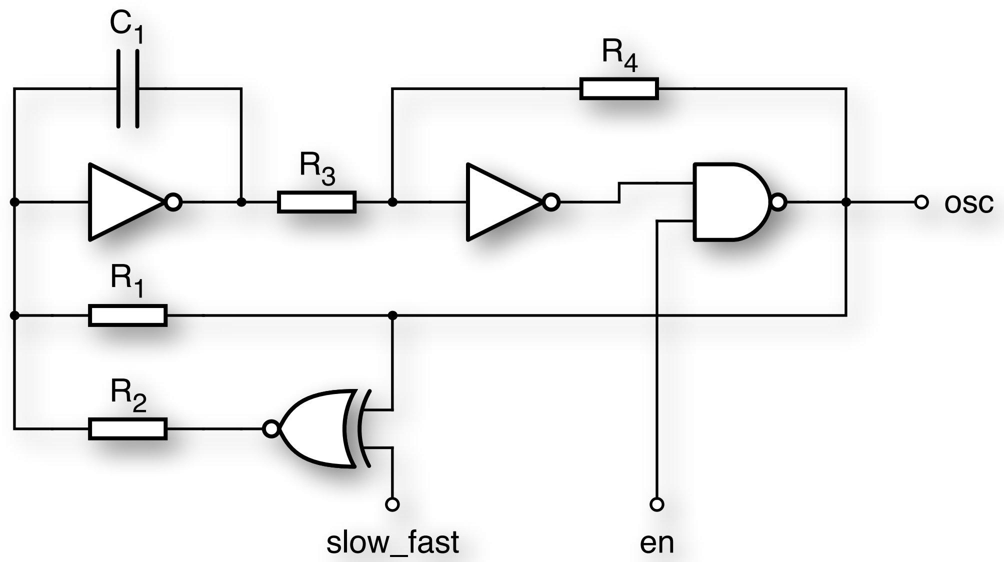
Two-strength inverter

Two-strength inverter:

when P_{sw} and N_{sw} are off, P_2 and N_2 make the inverter weaker (an therefore slower)

The two-strength inverter can be implemented as a standard cell thus making the full design a **digital semi-custom device**

Implementation of the two-speeds controlled oscillator: integrator and Schmitt trigger version



$$R_4 = 2 \cdot R_3$$

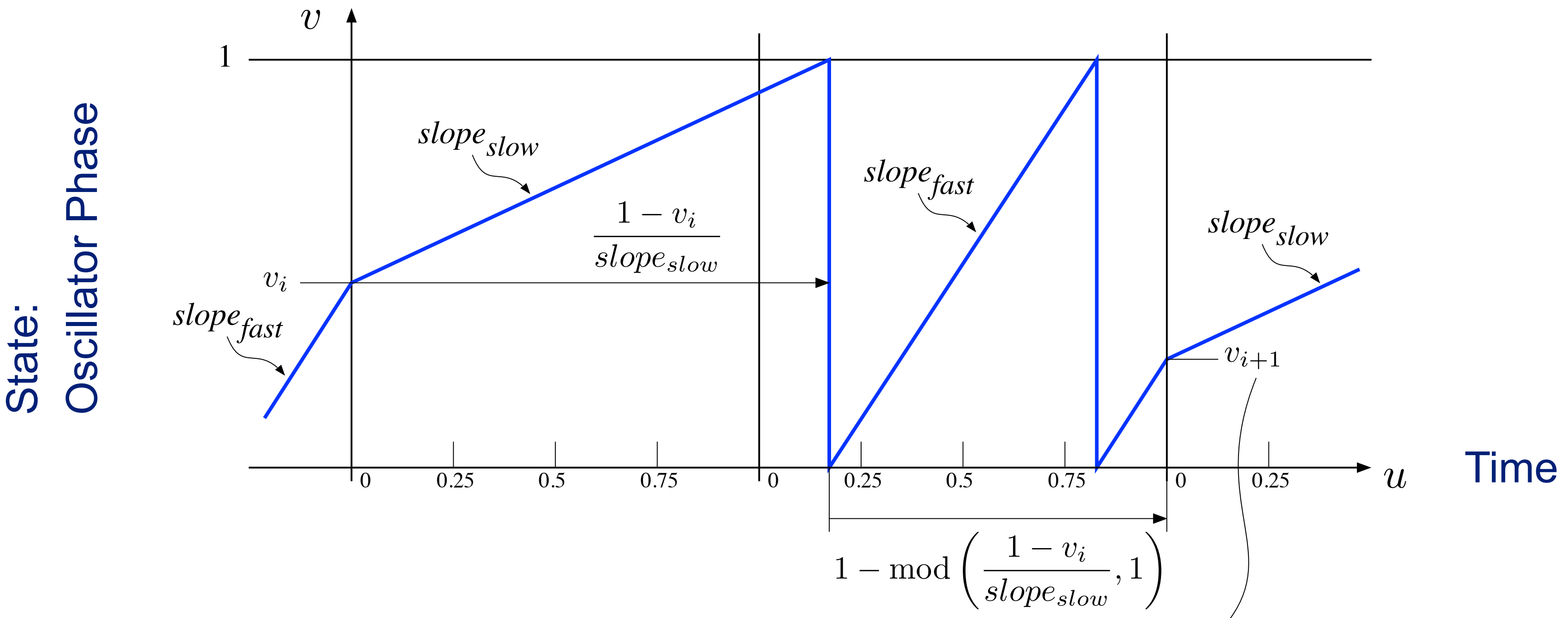
$$R_2 = R_1 \frac{k+1}{k-1}$$

$$C_1 = \frac{1}{f_{slow} \cdot R_1 \left(k+1\right)}$$

Just a “classical” triangular wave oscillator (integrator plus Schmitt trigger)

Depending on the slow_fast signal R1 and R2 operate in parallel or in counter-parallel thus changing the frequency between fast and slow

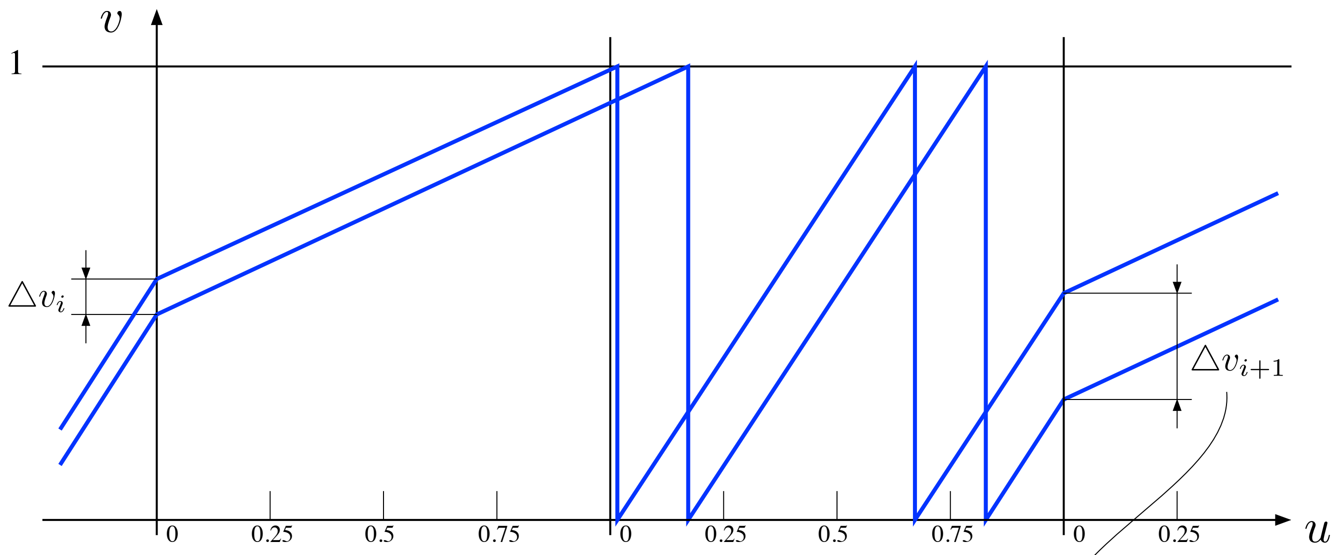
Discrete time representation: Chaotic Map



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

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

Evolution of two trajectories (jitter amplification mechanism): Lyapunov Exponent

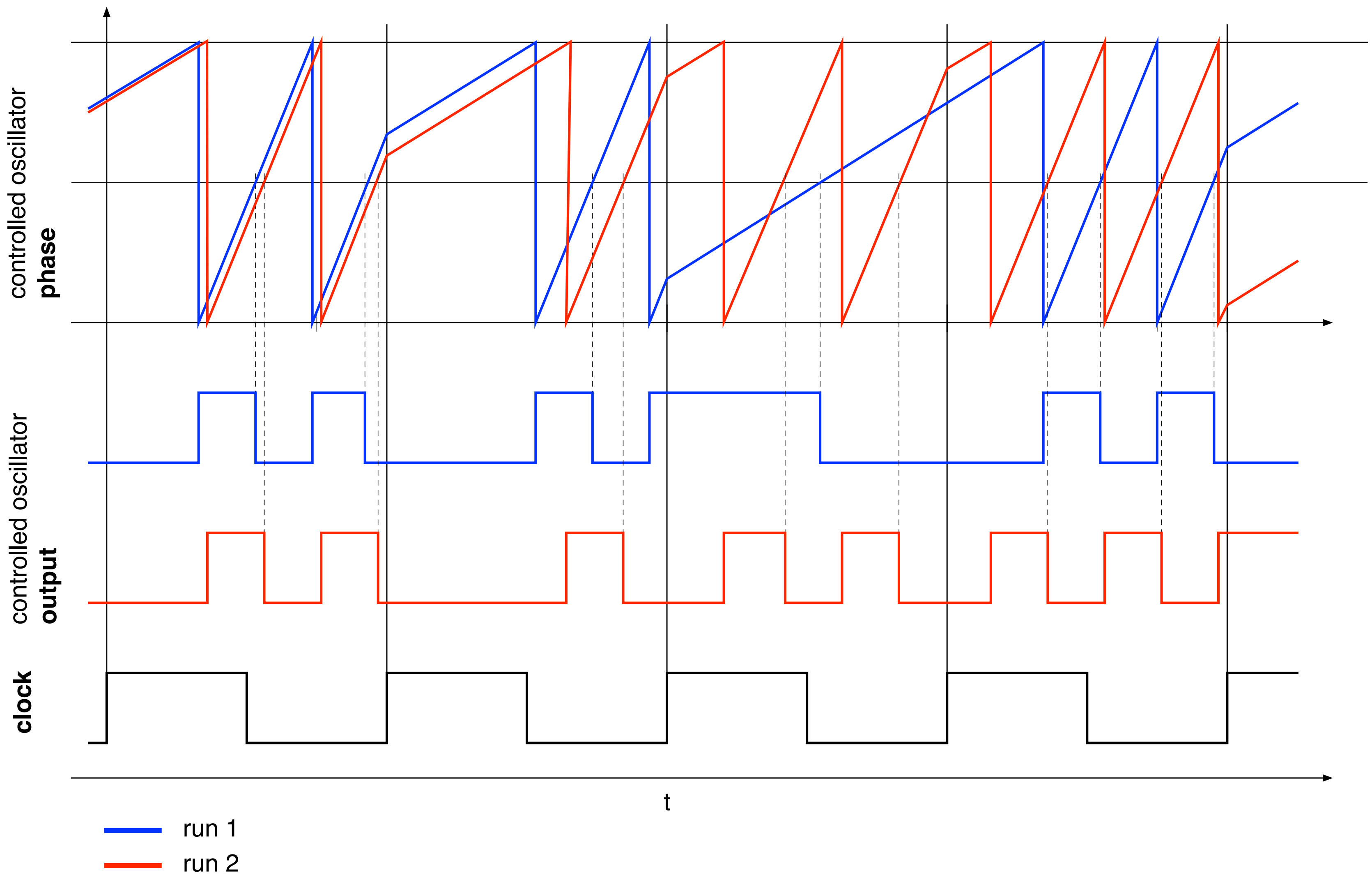


$$\triangle v_{i+1} = \frac{slope_{fast}}{slope_{slow}} \triangle v_i$$

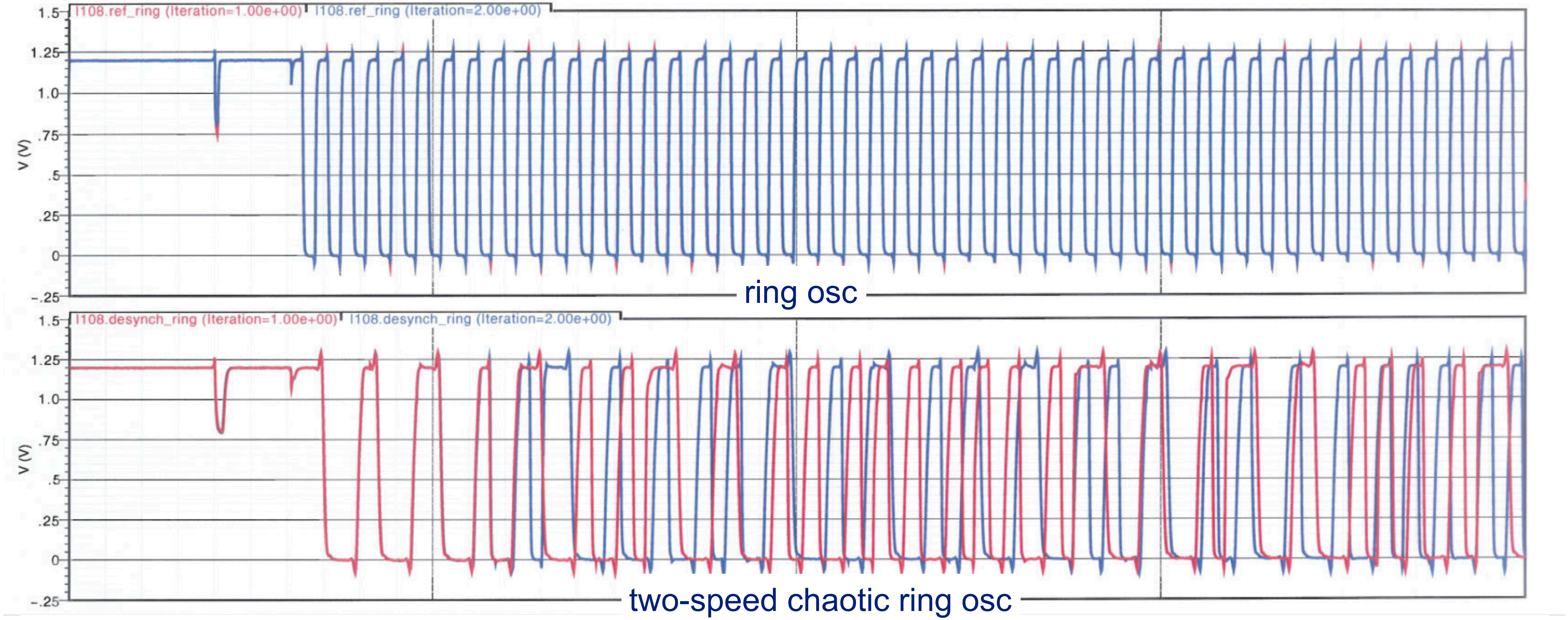
Noise (i.e. jitter) amplification results from the separation between trajectories which follows the law $\left|\delta v_{i+n}\right| = \left|\delta v_i\right| k^n$ where $k = slope_{fast}/slope_{slow}$

This is the evidence that the **Lyapunov exponent** is $\lambda = \ln \left|k\right|$

Two-speeds chaotic oscillator and clock traces over two runs

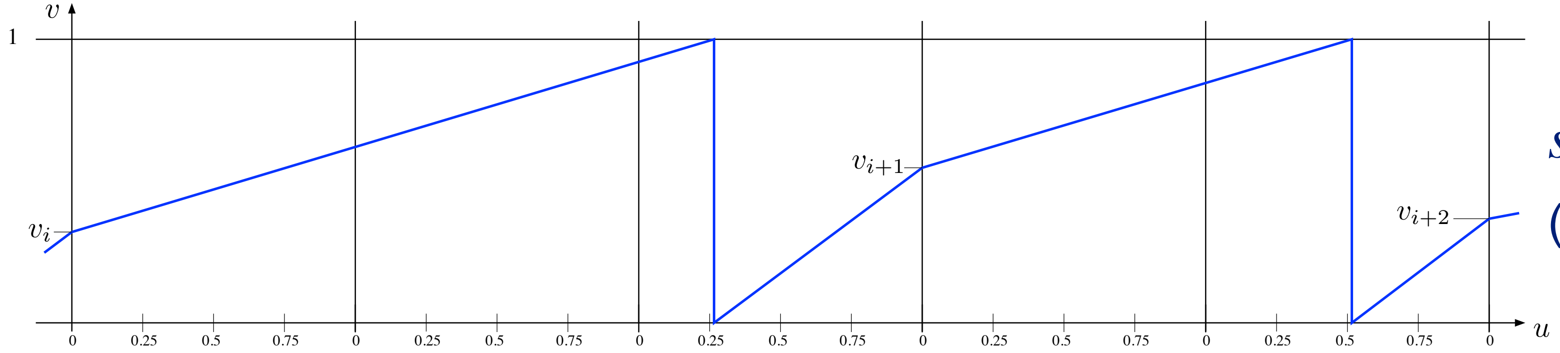


Transistor level simulation over two runs: conventional ring oscillator vs two-speeds chaotic ring oscillator



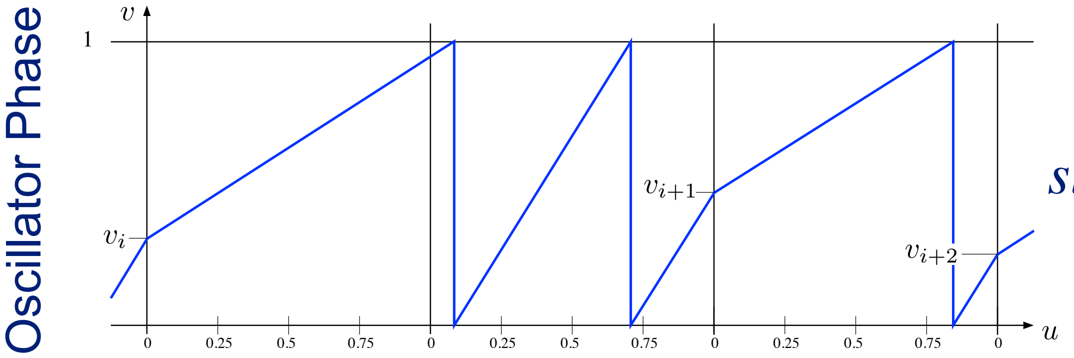
NOTICE: the intrinsic jitter of conventional ring oscillators (the ones normally used) is extremely poor. During operation, it is mainly due to almost deterministic environment disturbances (e.g. power supply).

Phase evolution for different clock vs controlled-oscillator frequencies

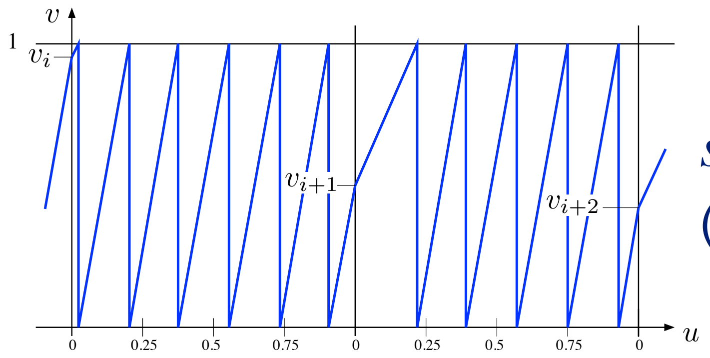


$slope_{fast} < 1$

(Fast is slower than Clock)



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



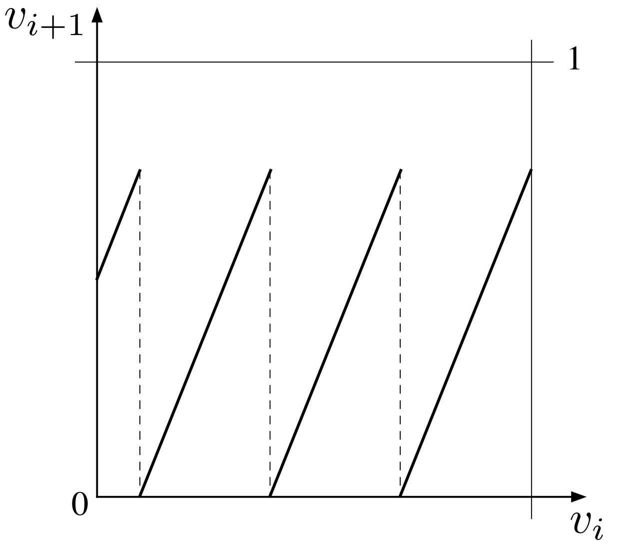
$slope_{slow} > 1$

(Slow is faster than Clock)

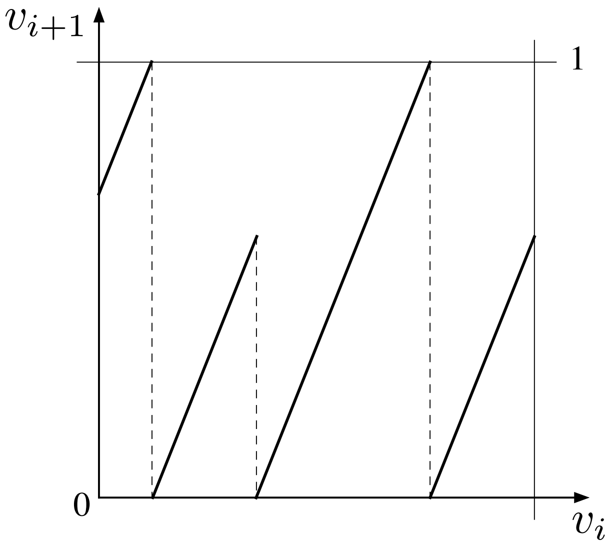
NOTICE: regardless of the clock frequency, a system iteration is always defined by two successive fast→slow transitions:

the system operates correctly in all conditions

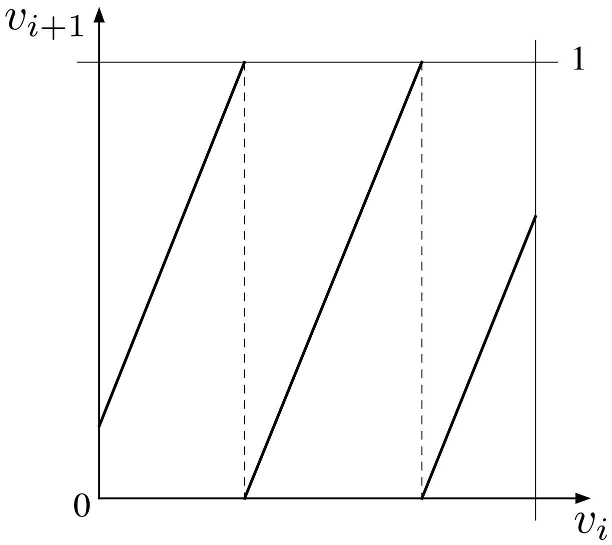
Chaotic map for different clock vs controlled-oscillator frequencies



$slope_{fast} < 1$



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



$slope_{slow} > 1$

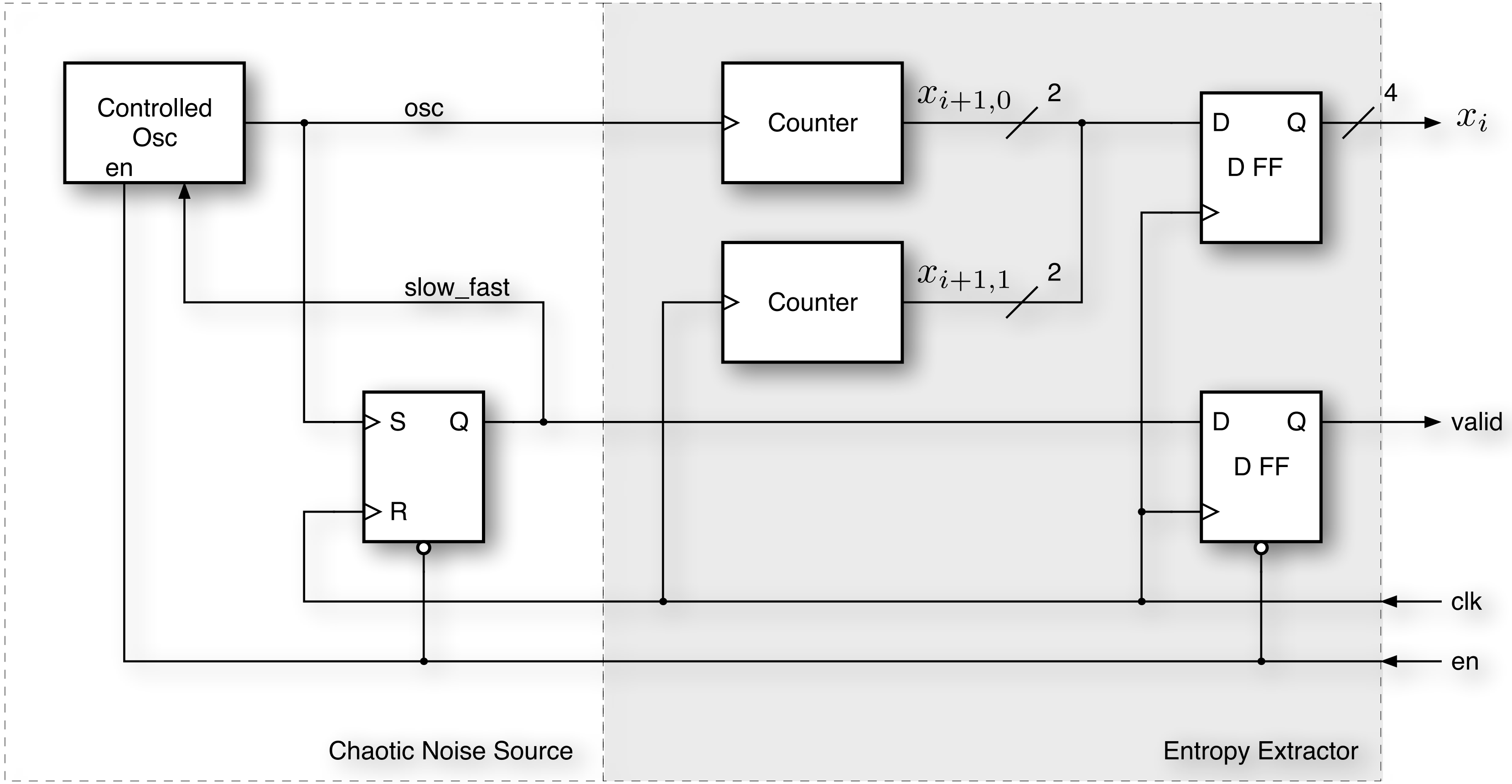
The chaotic map can assume three different shapes, but it is always a **piecewise linear map having constant derivative** $k = slope_{fast}/slope_{slow}$ and therefore a constant entropy rate $\log_2 |k|$

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

Outline

- Introduction
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - Invariant distribution, not a matter of noise
  - A didactical example
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - Chaotic oscillator
  - **Entropy extraction and results**
- Conclusions

Entropy extraction:
defining a generating partition of the state space



Each counter must feature a number of bits larger than the entropy rate

valid signal is generated at the end of each iteration (fast→slow transition). **No *valid* assertion in case of oscillator fault.**

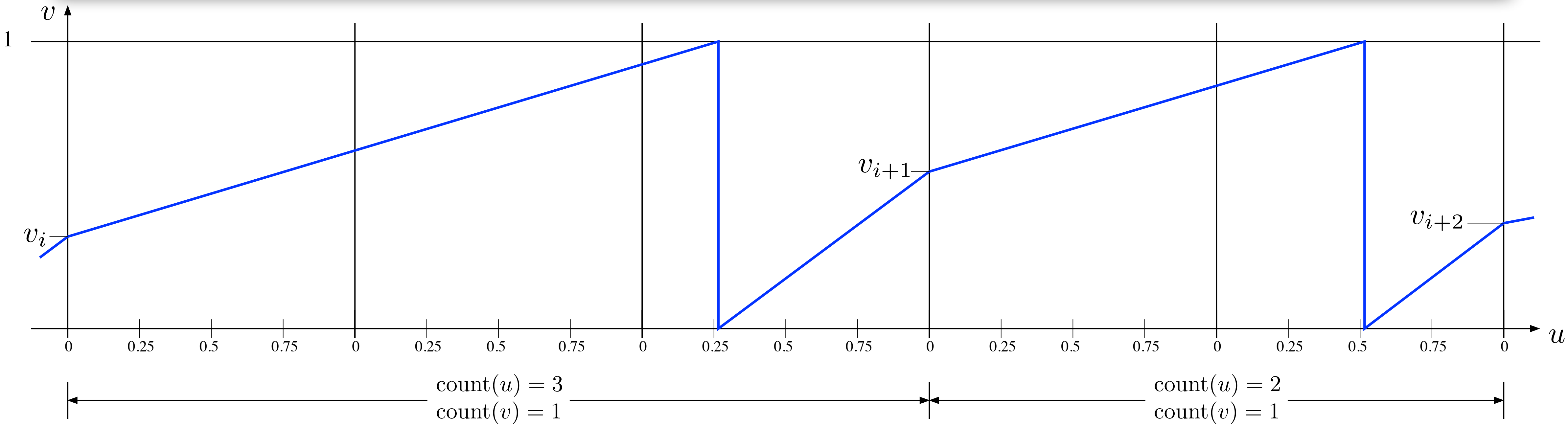
Regardless clock frequency, the entropy extractor collects the **full entropy rate** of the system by counting both the number of reference oscillator (i.e. clock) periods (u periods) and the number of controlled oscillator periods (v periods) which are executed during each iteration

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

The controlled oscillator $\boldsymbol{v}$ is always (i.e. also in fast mode) slower than the reference oscillator $\boldsymbol{u}$.

⇩

System iterations are always executed inside a single $\boldsymbol{v}$ (i.e. controlled oscillator) period.



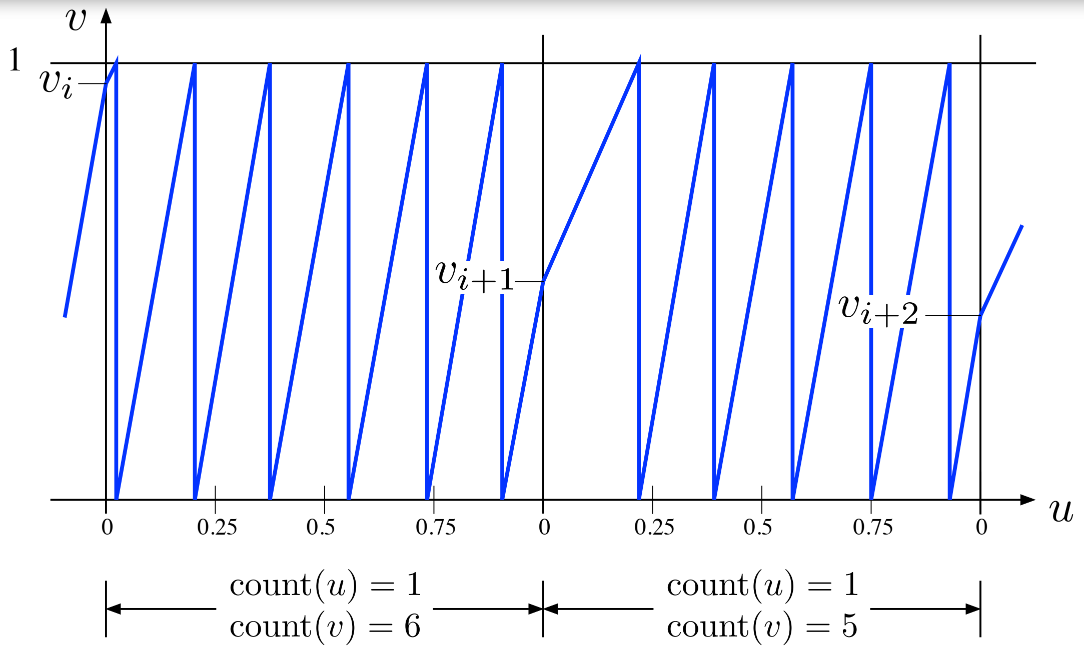
Entropy is extracted from the number of $\boldsymbol{u}$ (i.e reference oscillator) periods

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

The controlled oscillator $\boldsymbol{v}$ is always (i.e. also in slow mode) faster than the reference oscillator $\boldsymbol{u}$.

⇩

System iterations are always executed inside a single $\boldsymbol{u}$ (i.e. clock) period.

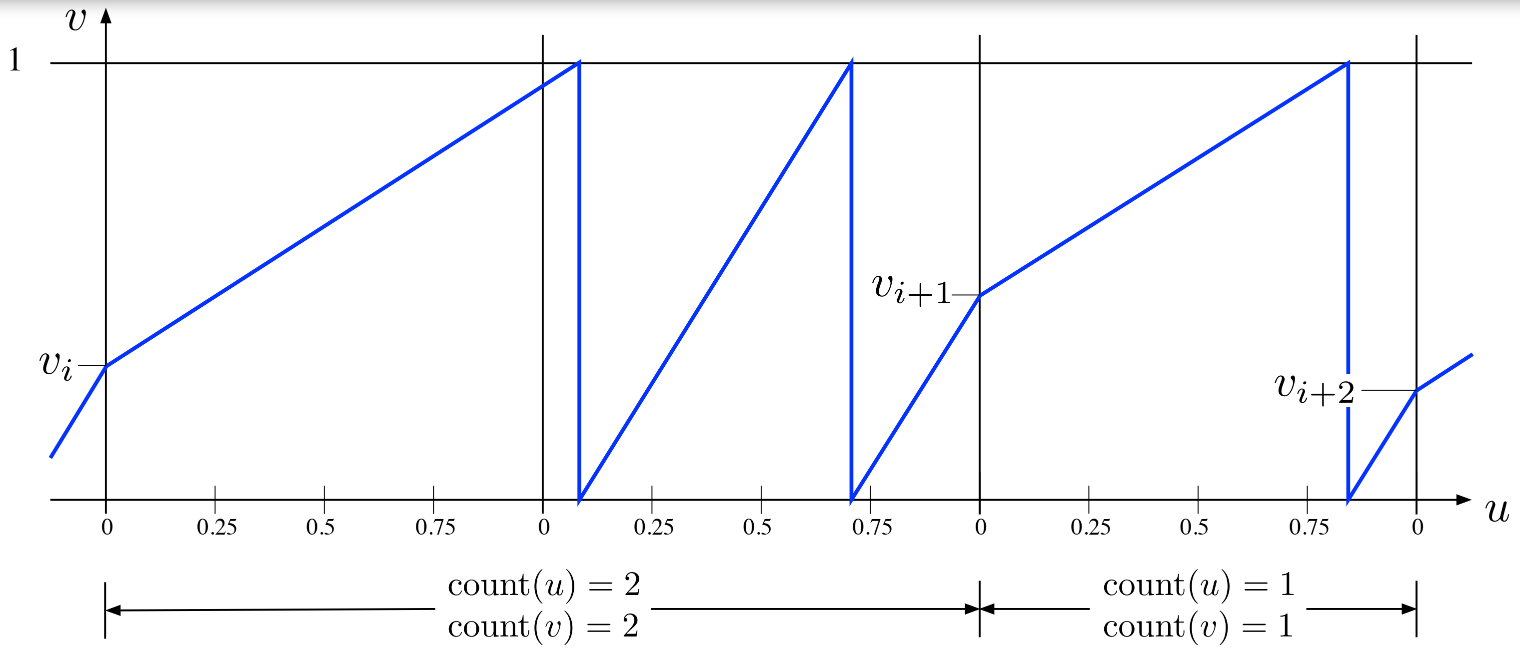


Entropy is extracted from the number of $\boldsymbol{v}$ (i.e controlled oscillator) periods

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

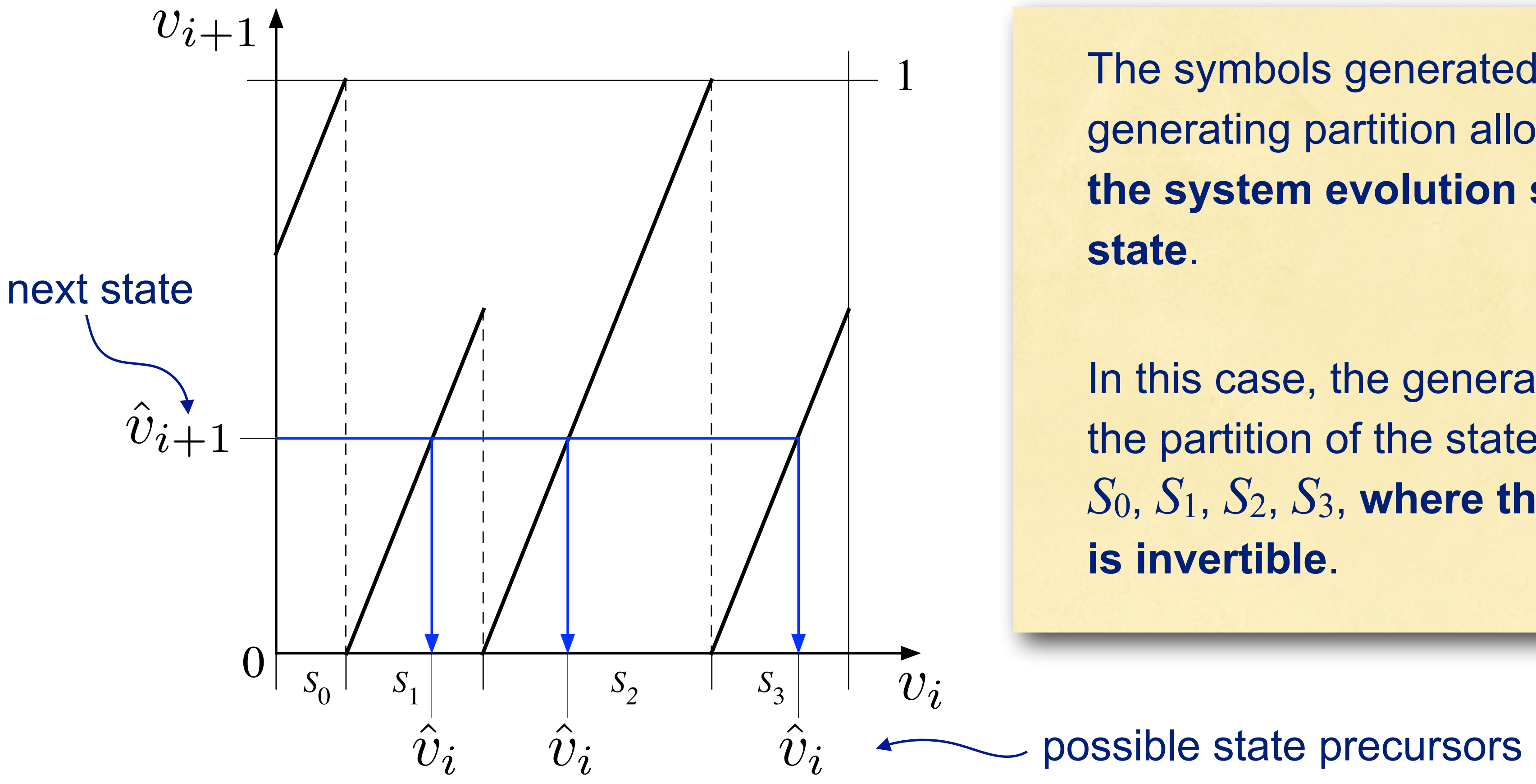
The controlled oscillator $\boldsymbol{v}$ can be slower or faster than the reference oscillator $\boldsymbol{u}$.

System iterations can include a different number of both reference $\boldsymbol{u}$ and controlled oscillator $\boldsymbol{v}$ periods.



Entropy is extracted from both the number of $\boldsymbol{u}$ (i.e. reference oscillator) and $\boldsymbol{v}$ (controlled oscillator) periods

Why all the system entropy is extracted?

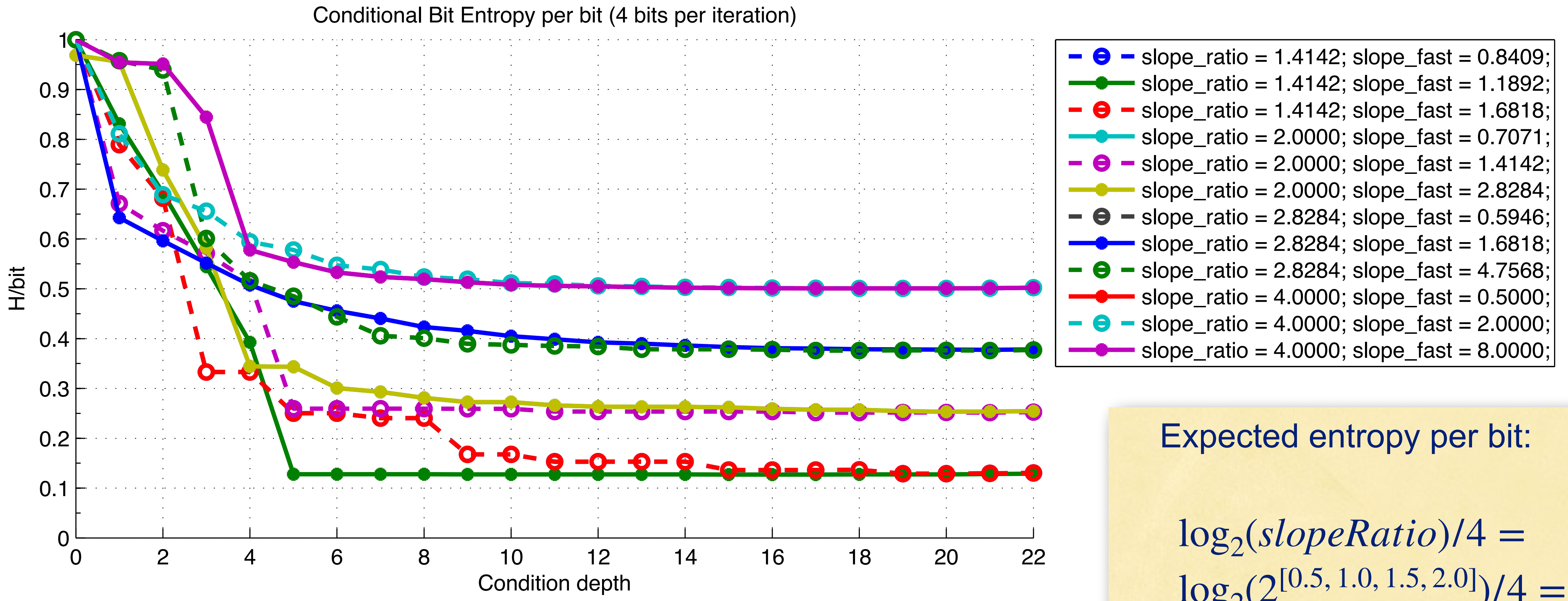


The symbols generated by means of a generating partition allows to **reverse (rewind) the system evolution starting from the current state**.

In this case, the 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 .

Conditional entropy estimation: simulation results ($slopeRatio = slope_{fast}/slope_{slow} = 2^{[0.5, 1.0, 1.5, 2.0]}$)

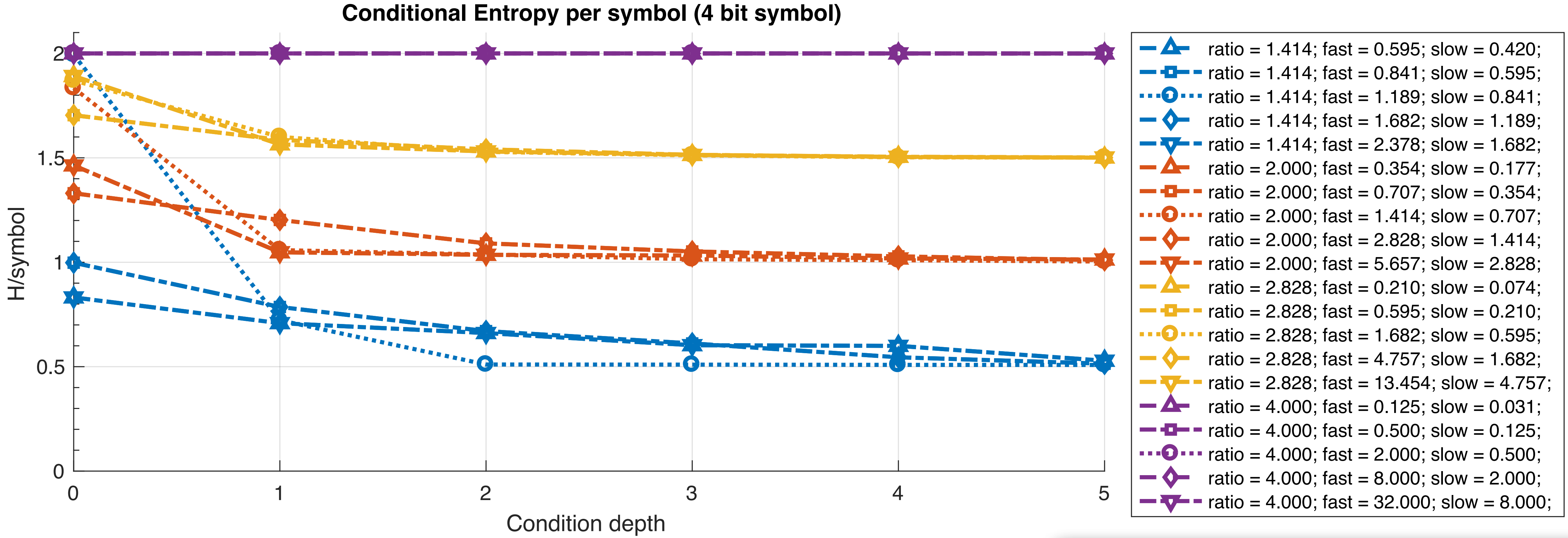


extractor: u, v count with reset;
n_sigma = 1.00e-06;
iterations = 4.00e+08;

Expected entropy per bit:

$$\log_2(slopeRatio)/4 = \log_2(2^{[0.5, 1.0, 1.5, 2.0]})/4 = [0.5, 1.0, 1.5, 2.0]/4 = [0.125, 0.250, 0.375, 0.5]$$

Conditional entropy estimation: simulation results ($slopeRatio = slope_{fast} / slope_{slow} = 2^{[0.5, 1.0, 1.5, 2.0]}$)



entropy extractor: with reset;
n_sigma = 1.00e-09;
test_length = 1.68e+07; (source iterations)

Expected entropy per bit:

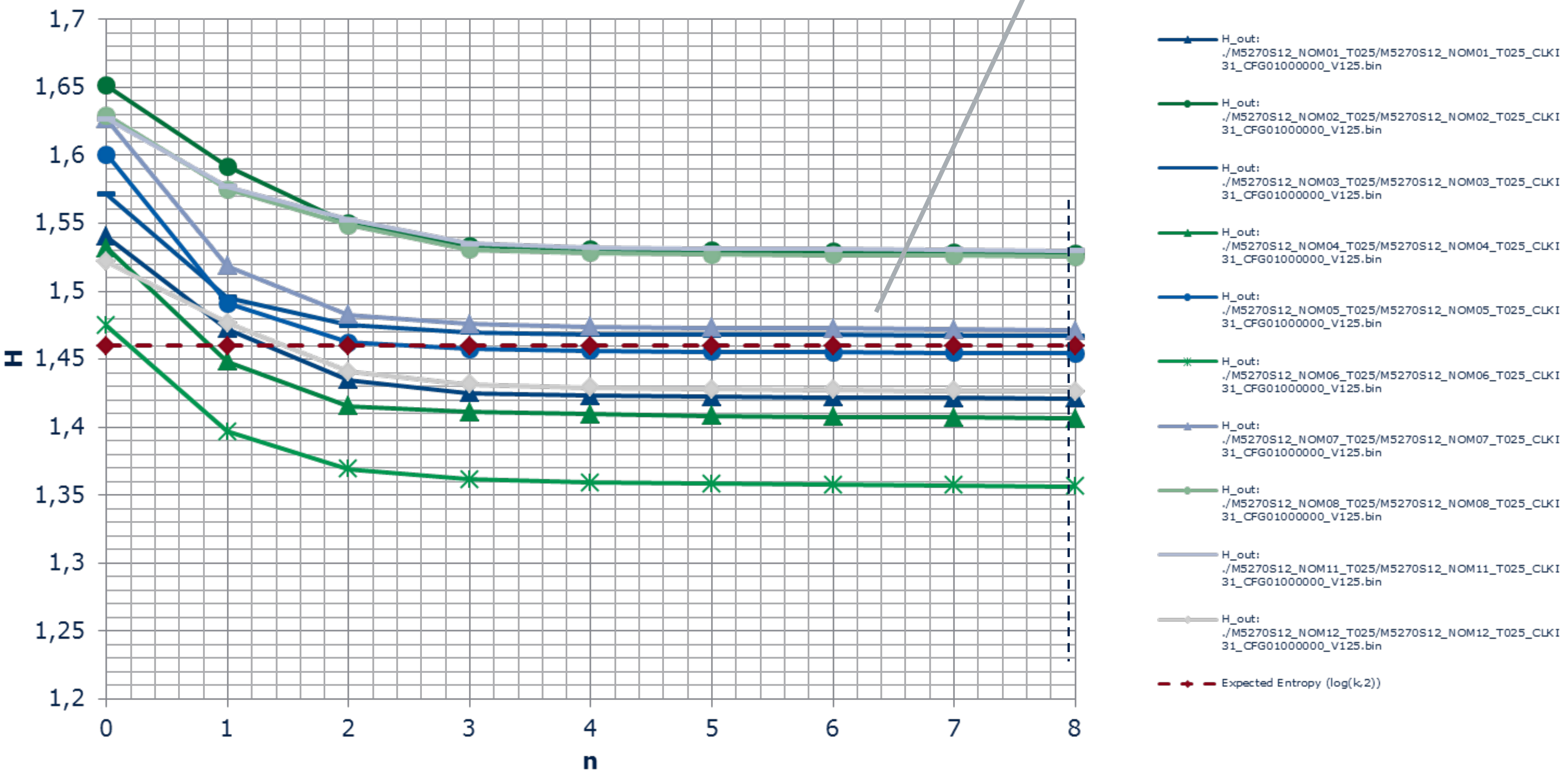
$$[0.125, 0.250, 0.375, 0.5]$$

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



Outline

- Introduction
- Some classic entropy source implementations
- Chaotic entropy source modelling
  - Stretching and folding, the power of exponential growth
  - Invariant distribution, not a matter of noise
  - A didactical example
  - Uncertainty expansion and entropy rate, also not a matter of noise
- An implementation example
  - Chaotic oscillator
  - Entropy extraction and results
- **Conclusions**

Conclusions

- For a well-designed chaotic system, entropy can be:
  - determined a priori
  - extracted completely
  - empirically verified a posteriori

- Implementation is simple and robust:
  - a chaos based RBG can be more than one order of magnitude more efficient of any P-RBG (because of high speed, just a simple, strong, hashing post-processing can be used)
  - no additional vulnerability with respect of a P-RBG (manipulations and/or faults)
  - correct redundant techniques can be applied (e.g. 4-8 sources), almost costless, instead of the usual ineffective and useless online tests

Simplicity is a solved complexity

Constantin Brâncuși

Romanian sculptor 1876 – 1957

References

Recommended reference for a rigorous and comprehensive approach to chaos theory:

- M. Cencini, F. Cecconi, A. Vulpiani, “Chaos: from simple models to complex systems”, World Scientific Publishing Company, (2009).

Comprensive description of the Random Bit Generator mentioned in this presentation (including generating partition and hashing post-processing):

- M. Bucci, R. Luzzi, “A fully-digital Chaos-based Random Bit Generator”, Festschrift for David Kahn LNCS volume 9100

Tutorial notes and answers to the most common questions and misunderstandings about chaotic systems:

- M. Bucci, “On the Expected Entropy Rate and Noise Model in a Discrete-time Uni-dimentional Piecewise Linear Chaotic System” (Feb. 2016)
- M. Bucci:, “Entropy Rate and Noise Model in Chaotic Entropy Sources - FAQ “ (Aug. 2018)