# True random number generation

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

*– John von Neumann (1951)*

*The generation of random numbers is too important to be left to chance.*
*– Robert R. Coveyou*

# True random numbers

<u>Not</u> Pseudo-Random Number Generator

- PRNG is algorithm operating on some seed
- TRNG measures physical state of random system

<u>Usual procedure</u>

- measurement of truly random system
- apply algorithm improving uniformity
- optional: then use as seed in PRNG

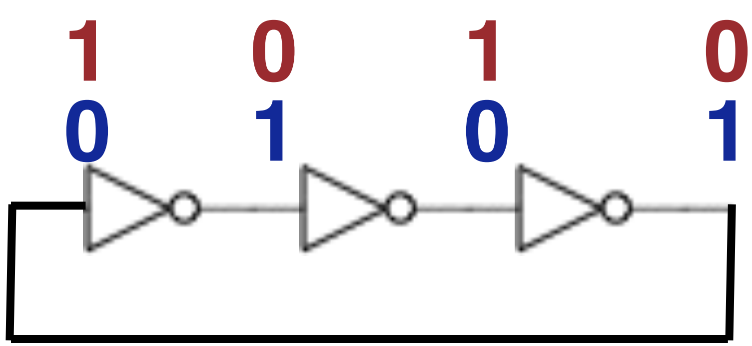
# Sources of randomness

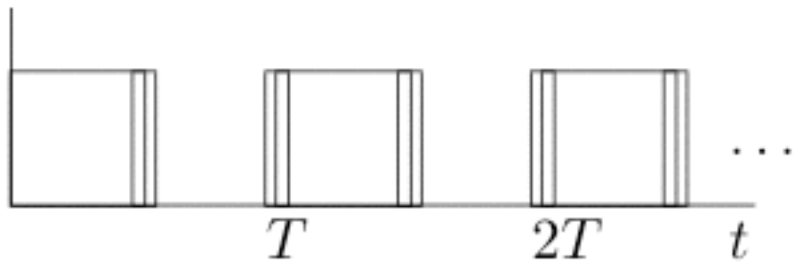
- noisy resistors
- ring oscillators
- avalanche diodes
- metastable flipflops
- antenna noise
- acoustic noise
- nuclear decay
- unstable lasers
- ...

# Ring oscillator

## Odd number of inverters in circular configuration

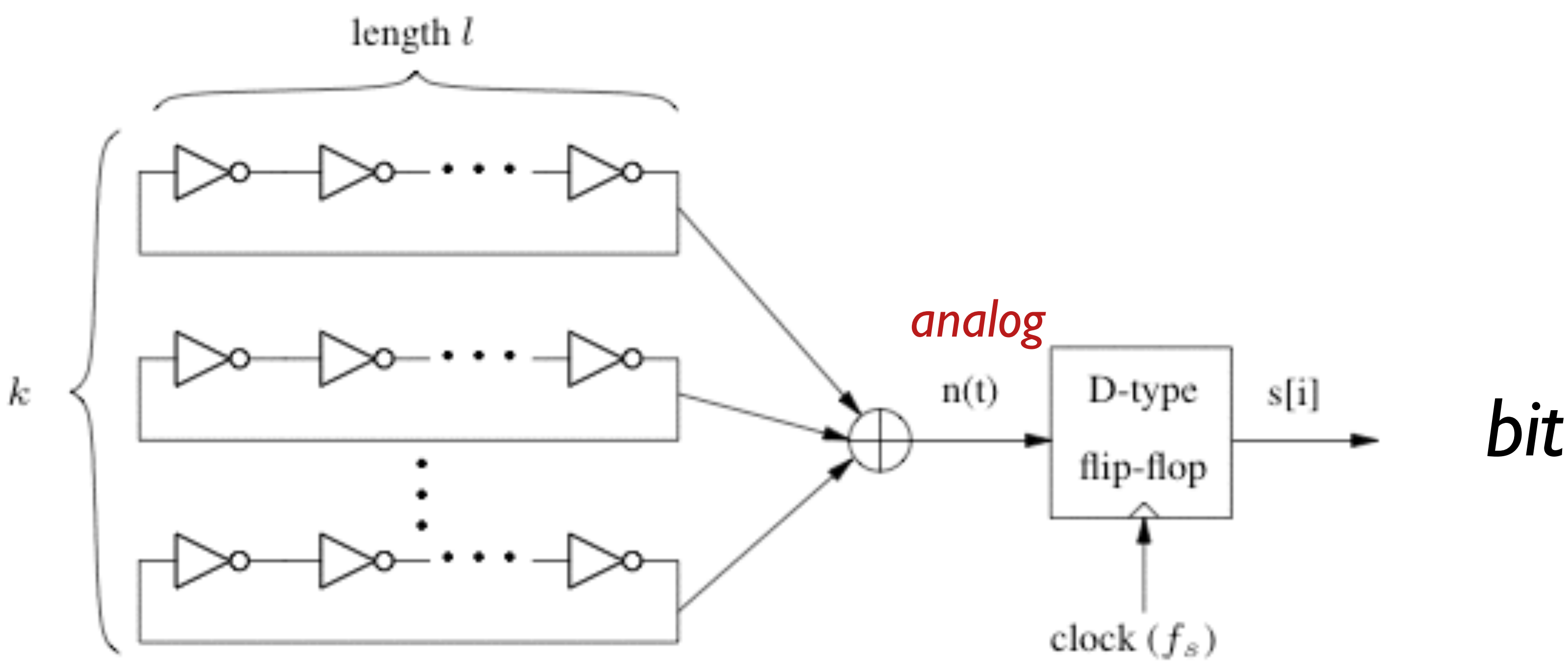
- conflicting logical state
- oscillation between 0/1 state
- propagation time partly random
- exact timing very sensitive to thermal noise
  - 'jitter'



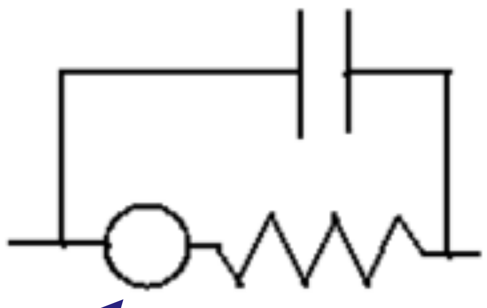

# Extracting randomness from jitter

- XOR several oscillators (analog operation)
- sample the analog signal
- force to logical 0 or 1
  - exact time of flank is Gaussian-distributed
- **fill rate** f: fraction of time where signal is unpredictable
  - can be tuned by choosing k, l



# Noisy resistor

Equivalent circuit
for resistor:



*voltage caused by thermal noise*

Amplitude has Gaussian distribution

$$\boxed{\langle V^2 \rangle = 4kT\,R\,\Delta f}$$

Johnson & Niquist, 1928

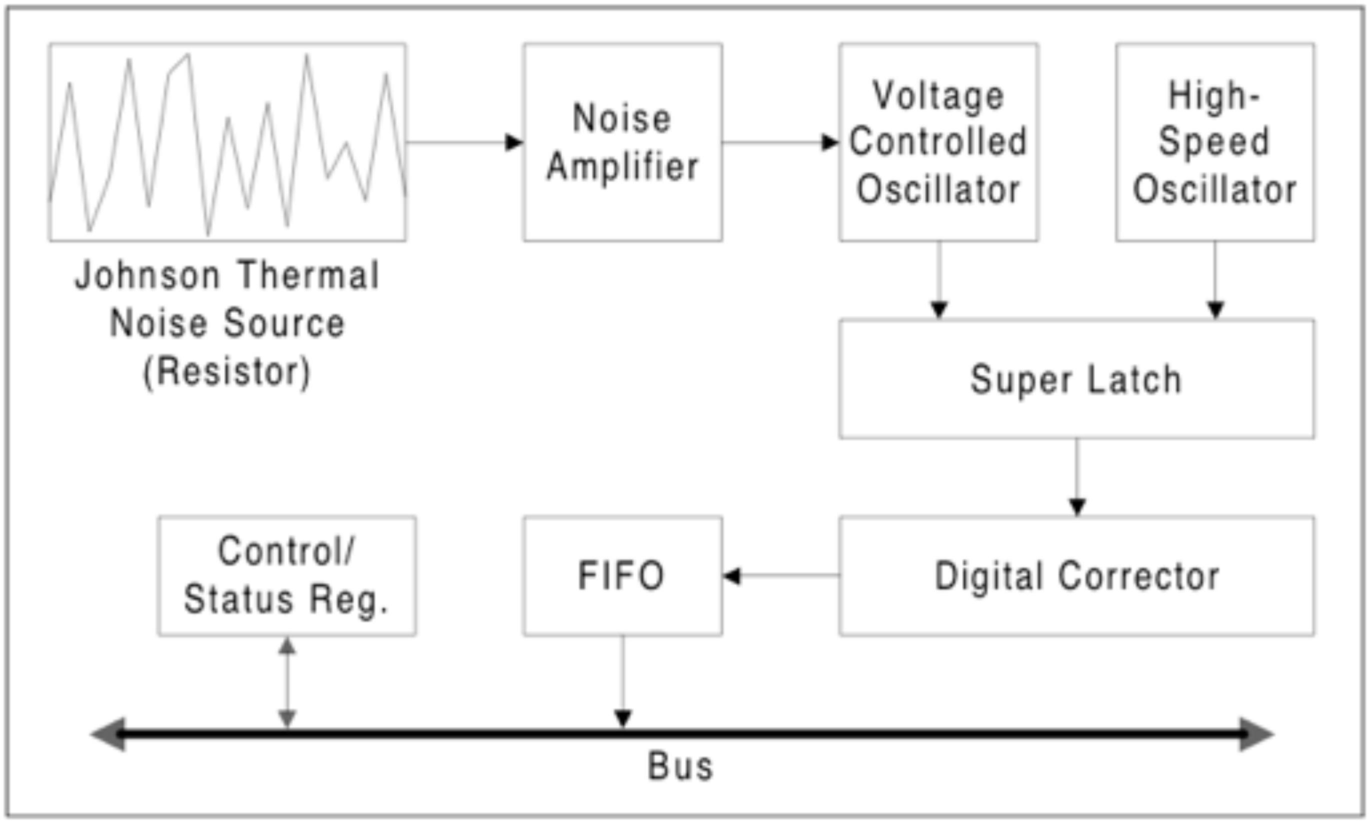
k = Boltzmann constant
T = temperature (Kelvin)
R = resistance
$\Delta$f = measured range of frequencies

# The Intel RNG

## Component of the Intel 80802 chip

*difference between two resistors*

[≈1999]




*Slow random clock drives measurements of fast clock*

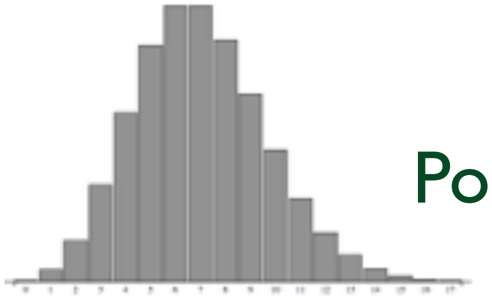
- improved version of von Neumann algorithm
- variable bit rate
- 75 Kbit/s after post-processing

# Radio-active decay

- Unstable atomic nucleus
- Exact moment of decay unpredictable
- $\lambda$ = prob of decay per time unit
- Pr[nucleus still exists] = $\exp(-\lambda t)$.
- Very tamper-proof!

Start with N nuclei;
Count #clicks in time $\Delta$t.

Pr[#clicks is *k*] = $e^{-N\lambda\triangle t}\frac{(N\lambda\triangle t)^k}{k!}$



Poisson distribution

# Algorithms for randomness extraction

Known continuous distribution f(x)

- generic procedure
- uses cumulative distr. function (cdf)

Known discrete distribution

- cdf + binning
- von Neumann algorithm
- piling it up: XOR-ing bits together
- resilient functions

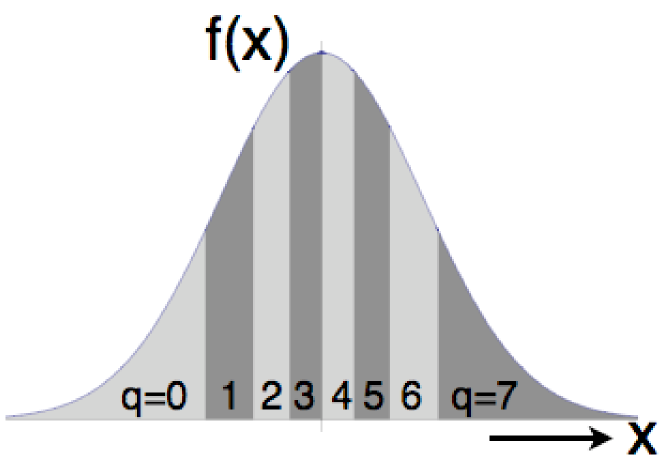
Unknown discrete distribution

- universal hash functions
- q-wise independent hashing

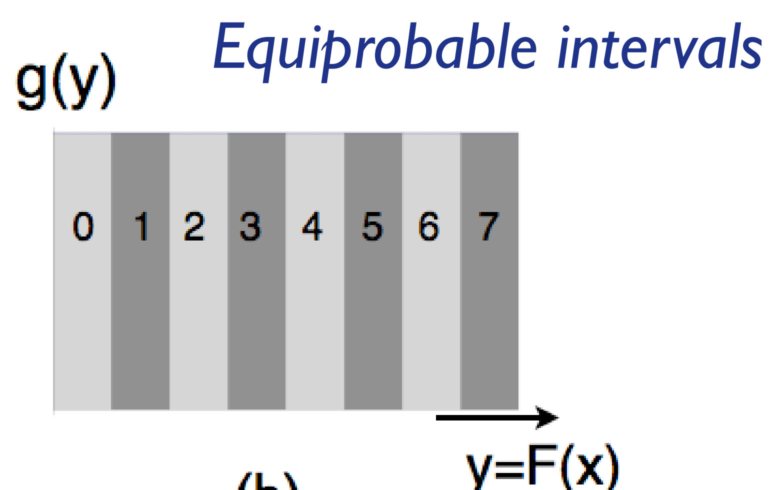
# Known continuous distribution

# Continuous random variables

- $X \sim f$
- Cumulative distribution function F
  - ❖ $\mathrm{Prob}[X<x] = F(x)$.
- The variable $Y := F(X)$ is uniform!




(a)




(b)

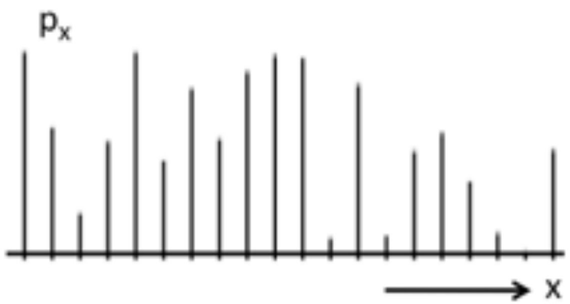
# Known discrete distribution

# Discrete random variables

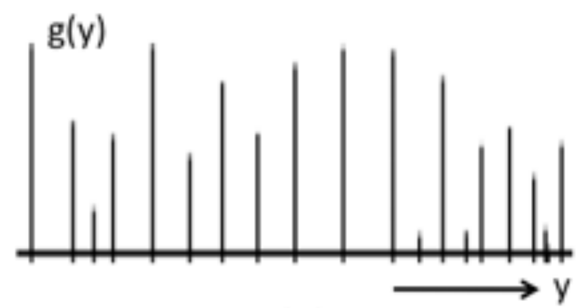
## Let’s try the cdf trick

$$f(x) = \sum_{i=1}^{n} p_i \delta(x - x_i)$$

$$Y = \sum_{i=1}^{n} p_i \Theta(X - x_i)$$



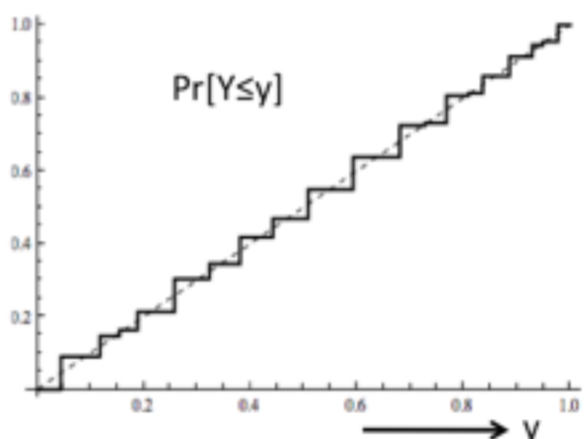

(a)



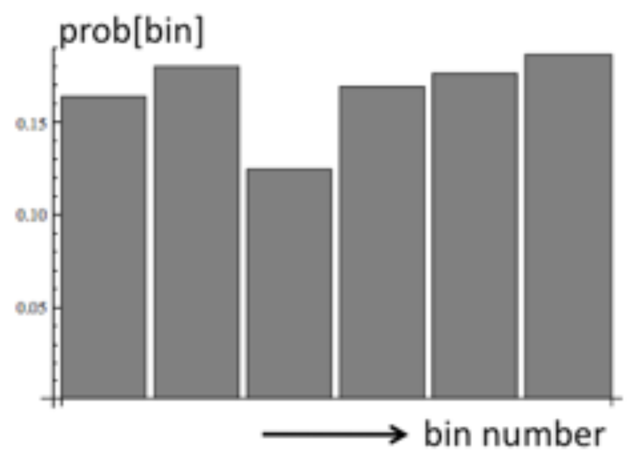

(b)

*not uniform!*

*Close to cdf of uniform distribution*




(c)




(d)

*not perfect, but it may do*

# von Neumann algorithm

Source = stream of bits from biased coin.

How to remove the bias?

Look at input pairs $(b_1, b_2)$

$$\begin{array}{rcl} b_1 = b_2 & : & \text{no output} \\ b_1 \neq b_2 & : & \text{output } b_1. \end{array}$$

| *b₁* | *b₂* | *output* |
|---|---|---|
| 0 | 0 | -- |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | -- |

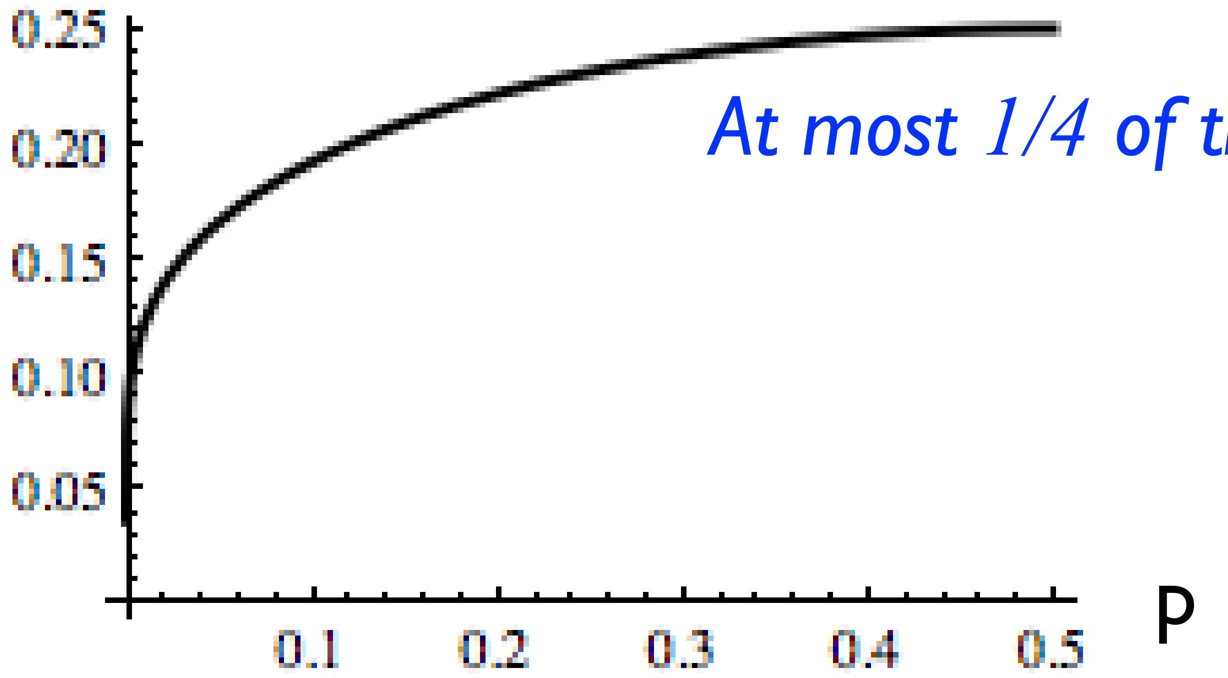
Question:

1. Why does this work?
2. How much entropy is lost?

# von Neumann algorithm: entropy loss

Fraction of retained entropy

$p(1-p)/h(p)$




*At most 1/4 of the entropy is kept*

# Improved von Neumann

| input | Neumann | improved |
|---|---|---|
| 0000 | - | - |
| 0001 | 0 | 00 |
| 0010 | 1 | 10 |
| 0011 | - | 0 |
| 0100 | 0 | 01 |
| 0101 | 00 | 00 |
| 0110 | 01 | 01 |
| 0111 | 0 | 01 |
| 1000 | 1 | 11 |
| 1001 | 10 | 10 |
| 1010 | 11 | 11 |
| 1011 | 1 | 11 |
| 1100 | - | 1 |
| 1101 | 0 | 00 |
| 1110 | 1 | 10 |
| 1111 | - | - |

- generates more bits
- less wasteful

# Enumeration of permutations

$x \in \{0,1\}^n$ containing t 1s

Assign a label $L$ to the permutation that turns $\underbrace{11\cdots1}_{t}\underbrace{0\cdots0}_{n-t}$ into $x$.

L is uniform on $\{0, \cdots, \binom{n}{t} - 1\}$.

Numerical example:
$n = 16$
$t = 5$ or $11$

Original entropy
$\approx 16\, h(5/16) \approx 14.3$

$$\binom{n}{t} = 4368 = 2^{12} + 272$$

$$L \in \{0, \ldots , 4367\}$$

If $L > 4095$: No output!
If $L < 4096$: Output binary representation of $L$.

This yields 12 perfect bits with prob. 4096/4368
and no output with prob. 272/4368.
→ On average 11.3 bits

# Piling up lemma

$$Y = X_1 \oplus X_2 \oplus \ldots \oplus X_n$$

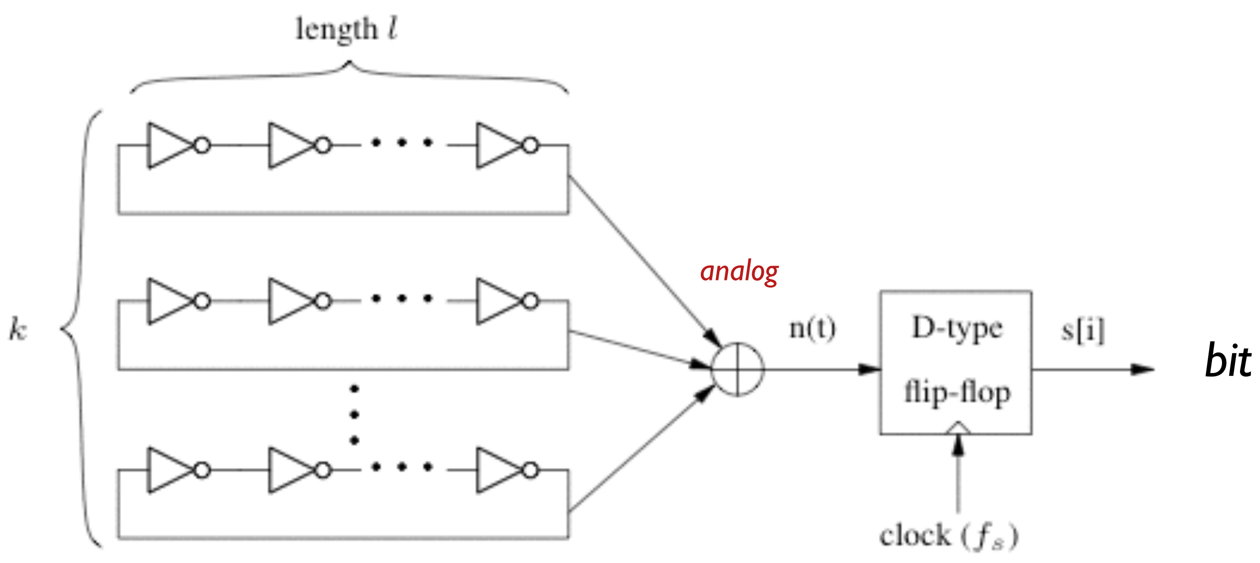
bias $\alpha_i = \Pr[X_i=1] - \Pr[X_i=0]$

$|\alpha_i| \leq 1$

Combined bias $\Pr[Y=1] - \Pr[Y=0] = (-1)^{n-1} \prod_i \alpha_i$

- reduced bias
- even one occurrence $\alpha_i=0$ already gives unbiased Y.
- but ... lots of entropy wasted

# Resilient functions




Expected:
*k* out of *n* bits predictable,
but ...
we don't know which ones!

<u>Post-processing</u>

- Apply ***resilient*** function $\Psi$ insensitive to *k* bits
- Def. of [n,m,k]-resilient:
  - $x \in \{0,1\}^n$, $\Psi(x) \in \{0,1\}^m$. Fix any k bits of x.
  - $\mathrm{Prob}[\Psi(X)=y \mid k \text{ bits of } X]$ is ***uniform*** on $\{0,1\}^m$.
- **Can be realized using error-correcting code**

# Unknown discrete distribution

# Worst case: Unknown distribution

Definition of a **strong extractor** "Ext"

for source min-entropy *m*, length $\ell$ and non-uniformity $\varepsilon$:

- Given a source X with $H_\infty(X) \geq m$
- uniformly drawn public randomness R
- $Z = \mathrm{Ext}(X, R) \in \{0,1\}^\ell$.

$$\mathbb{E}_r \Delta(Z|R = r;\; U_\ell) \leq \varepsilon$$

Uniform on $\{0,1\}^\ell$

In words:
Ext(X,R), *for known R*, is $\varepsilon$ away from uniform.

# Universal hash functions

Definition:

Universal family of hash functions $\{\Phi_r\}$

- Functions $\Phi_r$ from $\mathscr{X}$ to $\mathcal{T}$.
- Random seed R, uniformly chosen
- For any fixed x,x' with x' ≠ x:

$$\boxed{\mathrm{Prob}[\Phi_R(x) = \Phi_R(x')] \leq 1/|\mathcal{T}|}$$

---

Existence of univ. hash functions guarantees existence of strong extractors for certain parameter range!

We will see this in a couple of slides ...

# Almost-universal hash functions

**Definition 3.11 (Almost universal family of hash functions)** *Let $\eta \geq 0$ be a constant. Let $\mathcal{R}$, $\mathcal{X}$ and $\mathcal{T}$ be finite sets. Let $\{\Phi_r\}_{r \in \mathcal{R}}$ be a family of hash functions from $\mathcal{X}$ to $\mathcal{T}$. The family $\{\Phi_r\}_{r \in \mathcal{R}}$ is called $\eta$-almost universal iff, for $R$ drawn uniformly from $\mathcal{R}$, it holds that*

$$\mathrm{Prob}[\Phi_R(x) = \Phi_R(x')] \leq \eta$$

*for all $x, x' \in \mathcal{X}$ with $x' \neq x$.*

universal for $\eta = 1/\,|\mathcal{T}|$

# Leftover hash lemma

**Theorem 3.12 (Leftover hash lemma)** *Let $X \in \mathcal{X}$ be a random variable. Let $\delta \geq 0$ be a constant. Let $F : \mathcal{X} \times \mathcal{R} \to \{0,1\}^\ell$ be a $2^{-\ell}(1+\delta)$-almost universal family of hash functions, with seed $R \in \mathcal{R}$. Then*

$$\Delta(F(X,R)R;\ U_\ell R) \leq \frac{1}{2}\sqrt{\delta + 2^{\ell - \mathsf{H}_2(X)}}, \tag{3.17}$$

Distance of F(X,R) from uniformity, given R

*Proof is rather long, see appendix in lecture notes.*

*The $H_2$ is the Rényi entropy of order 2,*

$$H_2(X) = -\log \sum_x (p_x)^2$$

# When is extractor guaranteed to exist?

Forget about δ for the moment,

$$\Delta(F(X,R)YR;\ U_\ell YR) \le \frac{1}{2}\sqrt{\cancel{\delta +}\, 2^{\ell - \mathsf{H}_2(X|Y)}}$$

*Set this to ε*

If $\ell \le \mathsf{H}_2(X|Y) + 2 - 2\log\frac{1}{\varepsilon}$ then UHF gives Stat.dist ≤ ε

*Quality of the source*

*Penalty for demanding ε-uniformity*

**Lots of entropy wasted!**

# q-wise independent hashing

- Very recent result [Dodis et al. 2014].
- Limited use compared to UHF
  - MACs, signatures, keyed hashes

Definition similar to UHF:

**Definition 3.15** *A q-wise independent family of hash functions from $\mathcal{X}$ to $\mathcal{Y}$ is a set $\{h_s\}_{s\in\mathcal{S}}$ of functions $h_s : \mathcal{X} \to \mathcal{Y}$ with the following property,*

$$\forall_{\text{distinct } x_1,\ldots,x_q \in \mathcal{X}} \forall_{y_1,\ldots,y_q \in \mathcal{Y}} \quad \Pr[h_S(x_1) = y_1 \wedge \cdots \wedge h_S(x_q) = y_q] = |\mathcal{Y}|^{-q}. \tag{3.20}$$

# Result of q-wise independent hashing

Start with an algorithm that has "security δ" when used with a perfect key.

Compress to size $\ell \leq \mathsf{H}_\infty(X) - 4 - \log\log\frac{1}{\varepsilon}$

using $q = 6 + \lceil \log \frac{1}{\varepsilon} \rceil$.

Then the security of the algorithm goes from δ to

$$\delta' = 2\delta + \varepsilon.$$