Recent results on bit-flipping LDPC decoders

Chris Winstead[1,2], Gopalakrishnan Sundararajan[1], Emmanuel Boutillon[2]

[1] Department of Electrical and Computer Engineering
LE/FT Lab
Utah State University

[2] Lab-STICC
Université de Bretagne Sud

April 23, 2014

Introduction: Error Correcting Codes

Error Correcting Codes (ECC) improve the reliability of many electronic systems.

They are essential for communication and storage applications:

- Wireless network connections.
- High-speed wired and optical links.
- Satellite communications.
- Disk drives, memories and optical storage.

High-performance ECC schemes are complex; expensive to implement.

This presentation is about <u>tradeoffs</u> between complexity and performance.

Outline

1. Introduction to ECC
    1. Basic theory — practical issues and ultimate limits.
    2. LDPC Codes — structure and ultimate performance.

2. Bit-flipping algorithms
    1. Sub-optimal LDPC decoding methods.
    2. Details of a new method: Noisy Gradient Descent [1].
    3. Performance results and complexity analysis.

3. Tradeoff analysis and conclusions
    1. Ultimate energy/performance tradeoffs [2].
    2. Potential for noise-enhanced computation [3].
    3. Remaining problems in suboptimal decoding.

Outline

1. Introduction to ECC
   1. Basic theory — practical issues and ultimate limits.
   2. LDPC Codes — structure and ultimate performance.

2. Bit-flipping algorithms
   1. Sub-optimal LDPC decoding methods.
   2. Details of a new method: Noisy Gradient Descent [1].
   3. Performance results and complexity analysis.

3. Tradeoff analysis and conclusions
   1. Ultimate energy/performance tradeoffs [2].
   2. Potential for noise-enhanced computation [3].
   3. Remaining problems in suboptimal decoding.

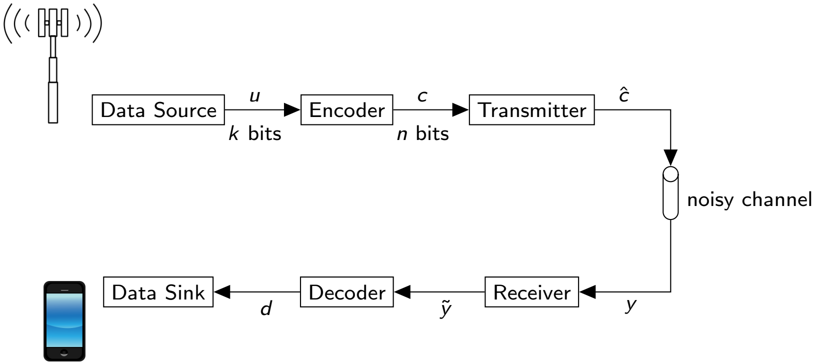
Outline

1. Introduction to ECC
   1. Basic theory — practical issues and ultimate limits.
   2. LDPC Codes — structure and ultimate performance.

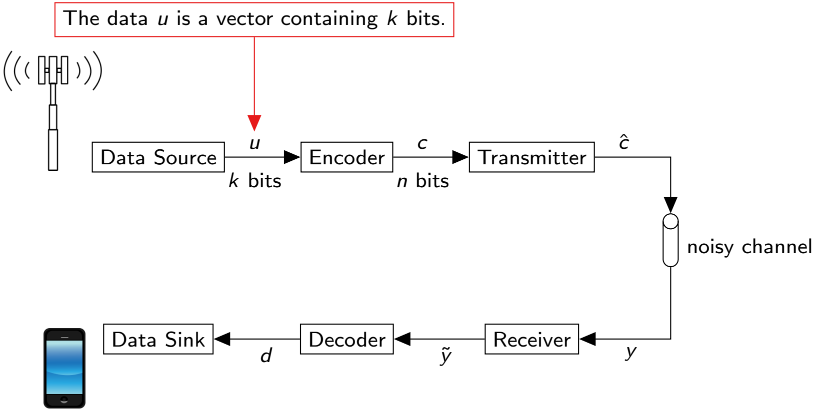
2. Bit-flipping algorithms
   1. Sub-optimal LDPC decoding methods.
   2. Details of a new method: Noisy Gradient Descent [1].
   3. Performance results and complexity analysis.

3. Tradeoff analysis and conclusions
   1. Ultimate energy/performance tradeoffs [2].
   2. Potential for noise-enhanced computation [3].
   3. Remaining problems in suboptimal decoding.

Introduction: Error Correcting Codes

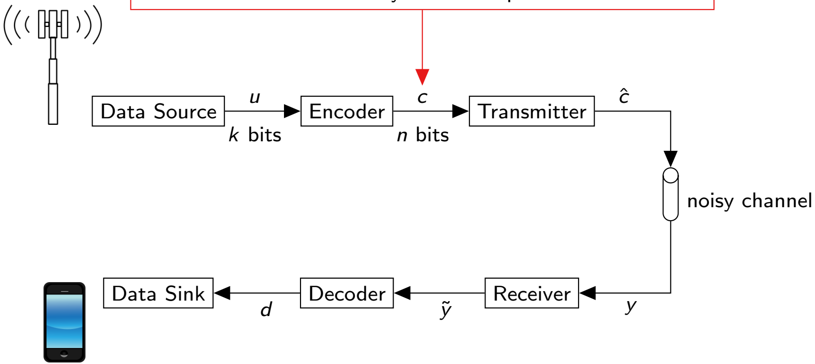


Introduction: Error Correcting Codes

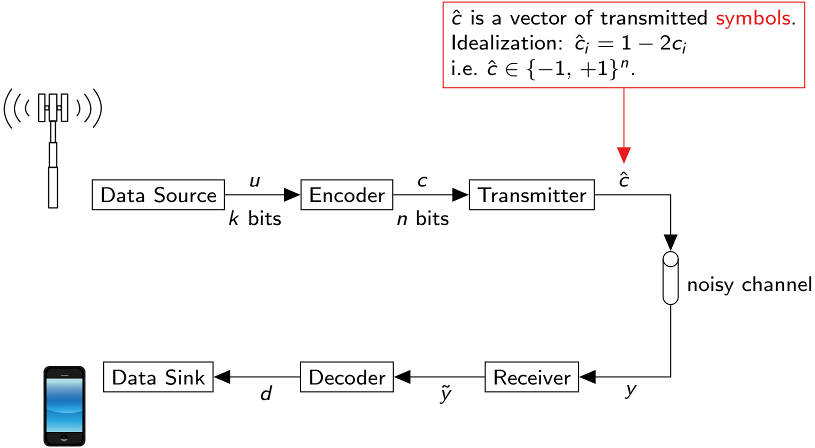


Introduction: Error Correcting Codes

The codeword c is a vector containing n bits, $n > k$.
The extra bits are called parity bits.
The encoder is described by a matrix operation: $c = u \times G$.



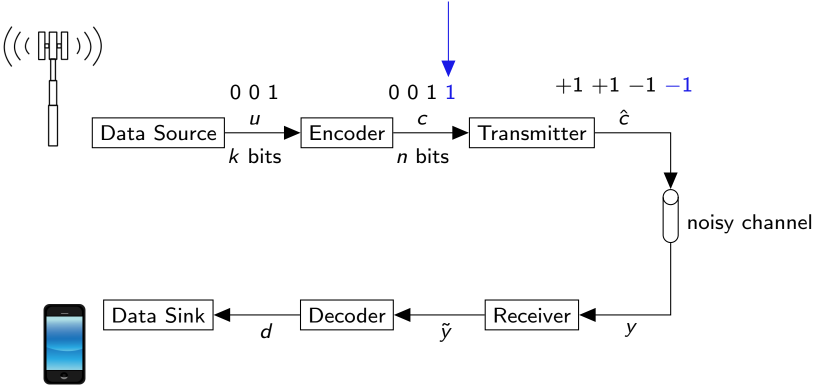
Introduction: Error Correcting Codes



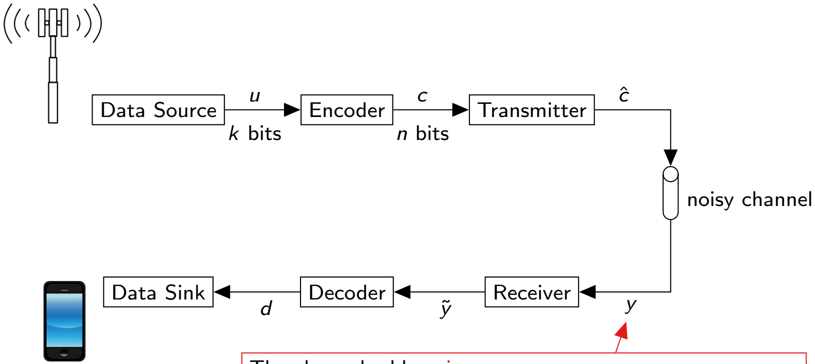
Introduction: Error Correcting Codes

Example:

Parity bit: $p = \sum_i c_i \mod 2$



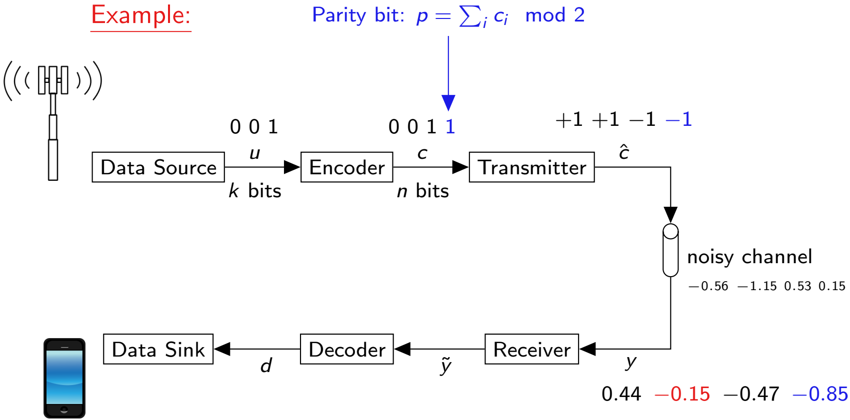
Introduction: Error Correcting Codes



The channel adds noise.
Idealization: Additive White Gaussian Noise (AWGN),
every $y_i = x_i + n_i$, where n_i is a Gaussian random sample.

Introduction: Error Correcting Codes

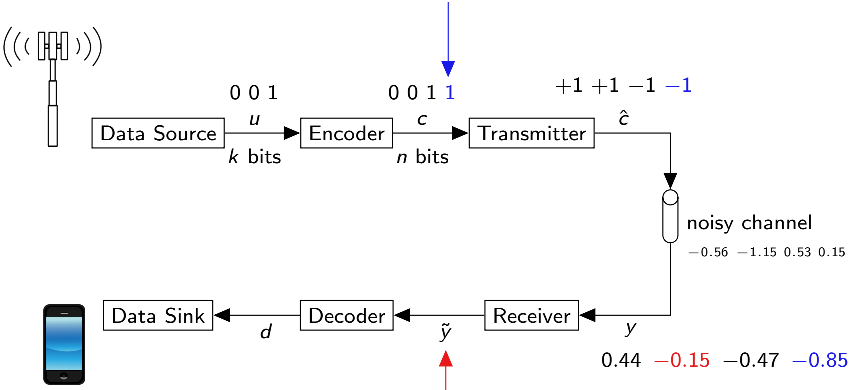
Example:



Introduction: Error Correcting Codes

Example:

Parity bit: $p = \sum_i c_i \mod 2$

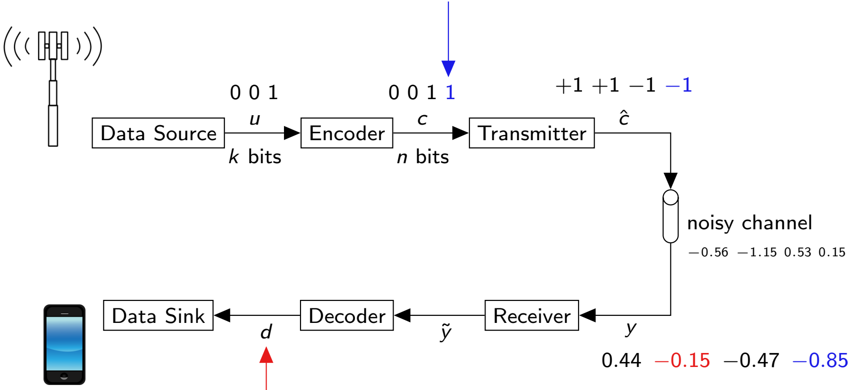


The receiver performs quantization.
Hard Information: $\tilde{y} = +1\ -1\ -1\ -1$
Soft Information: $\tilde{y} = 0.4\ -0.2\ -0.5\ -0.9$

Introduction: Error Correcting Codes

Example:

Parity bit: $p = \sum_i c_i \mod 2$

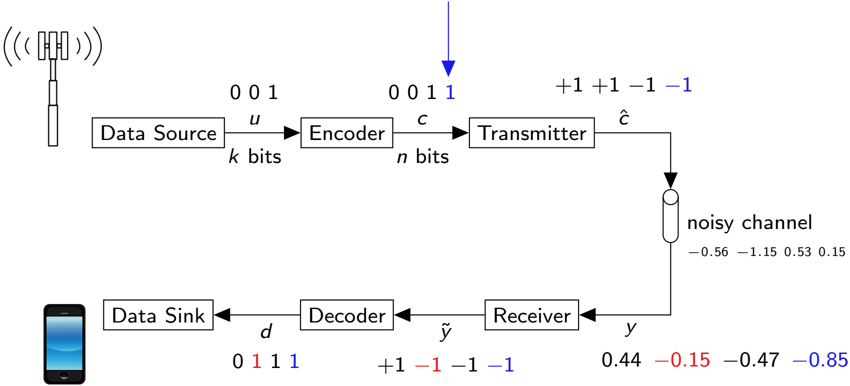


The decoder tries to estimate c from $\tilde{y}$.
Decoding is successful when $d = c$.

Introduction: Error Correcting Codes

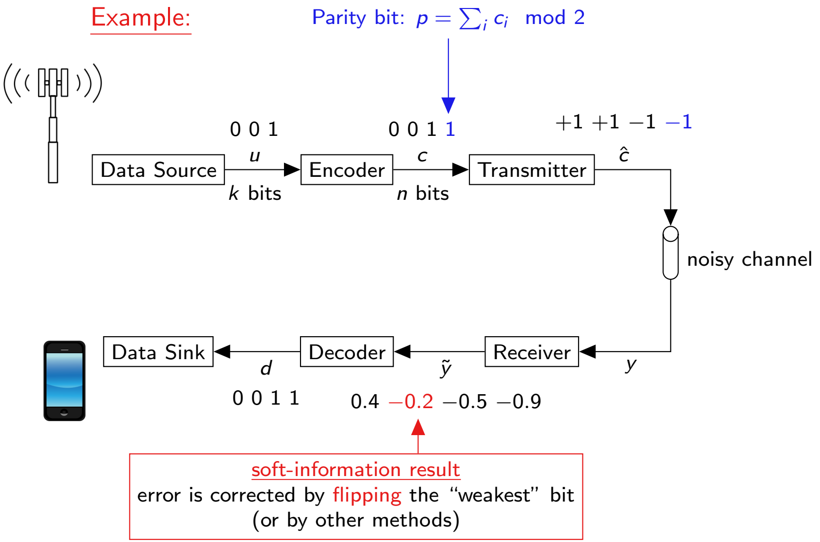
Example:

Parity bit: $p = \sum_i c_i \mod 2$



hard-information result
error is not correctable

Introduction: Error Correcting Codes



Evaluating and Comparing Decoders

Decoder performance is measured by the Bit Error Rate (BER).

BER is a function of Signal-to-Noise Ratio (SNR) at the receiver:

$$\mathrm{SNR} \triangleq \frac{E_b \text{— Signal power, energy per bit}}{N_0 \text{— Noise power spectral density}}$$

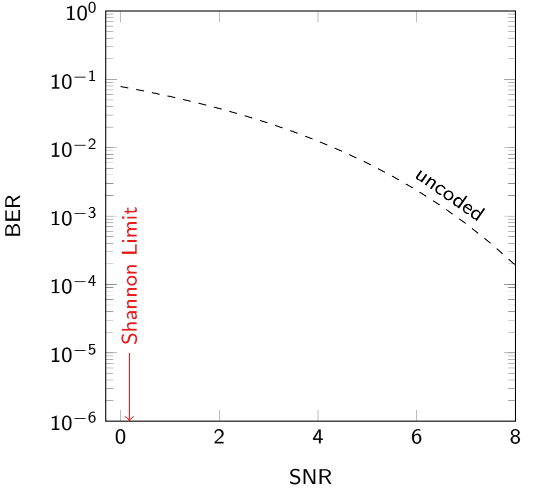
Usually SNR is expressed in dB:

$$\mathrm{SNR\ (dB)} = 10 \log_{10} \left(\frac{E_b}{N_0} \right)$$

Lastly the effective SNR depends on the code's Rate $R \triangleq k/n$. In our idealization, $E_b = 1/R$, so

$$SNR = 10 \log_{10} \left(\frac{1}{RN_0} \right)$$

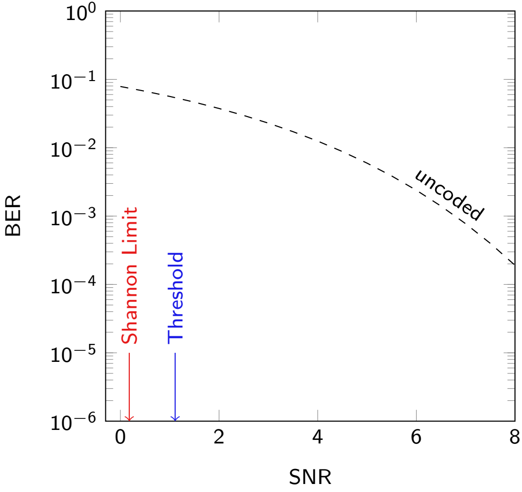
Evaluating and Comparing Decoders



For a specific rate, say $R = 0.5$, Shannon theory tells us the absolute minimum SNR.

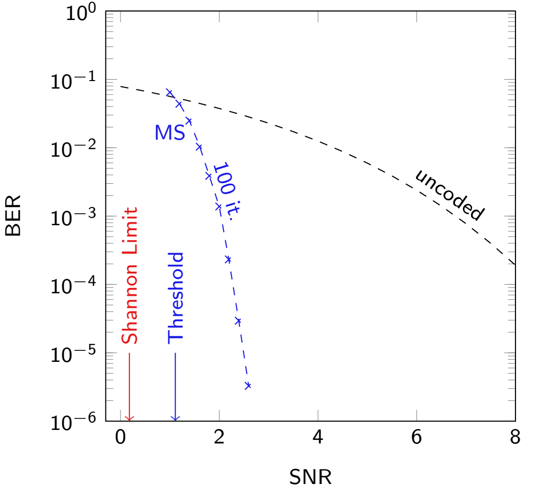
Turbo Codes [4, 5, 6] and LDPC Codes [7, 8, 9] are practical solutions that can come close to the Shannon limit.

Evaluating and Comparing Decoders



For a particular family of LDPC codes and decoding algorithms, we can also obtain a code-specific threshold indicating the limit for this code [10, 11, 12].

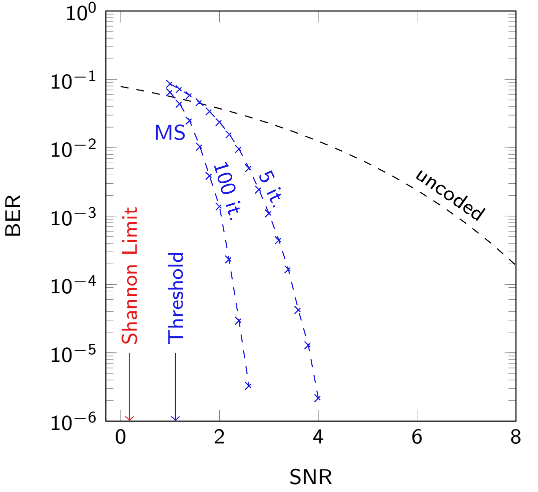
Evaluating and Comparing Decoders



High-performance algorithms, like Belief Propagation (BP), come closest to the threshold.

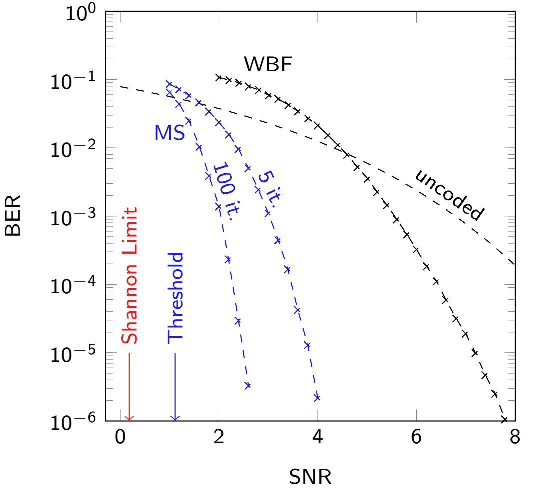
Approximate algorithms, like Min-Sum (MS), are fairly close to BP [13, 14].

Evaluating and Comparing Decoders



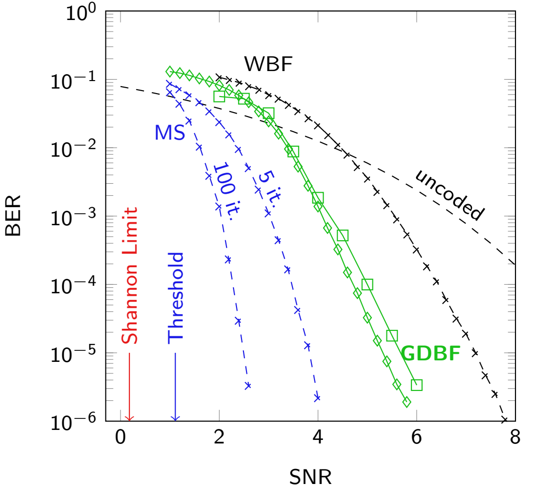
Decoding algorithms are iterative, meaning they require a large number of repeated calculations. In practice, we can trade between performance and complexity by operating with fewer iterations.

Evaluating and Comparing Decoders



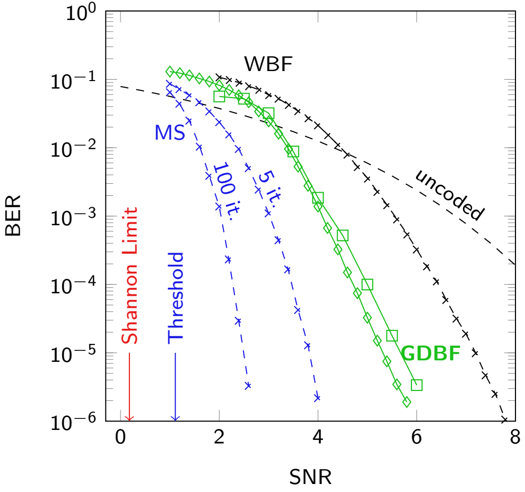
Alternative: so-called Weighted Bit-Flipping (WBF) algorithms have extremely low complexity, but with a large penalty in performance [15, 16].

Evaluating and Comparing Decoders



Numerous bit-flipping algorithms have been devised to improve performance. Gradient Descent Bit-Flipping (GDBF) algorithms provide a good balance between performance and complexity [17].

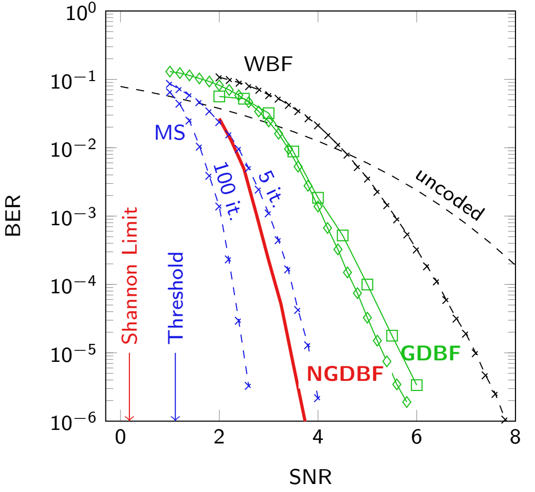
Evaluating and Comparing Decoders



Numerous bit-flipping algorithms have been devised to improve performance. Gradient Descent Bit-Flipping (GDBF) algorithms provide a good balance between performance and complexity [17].

Some bit-flipping algorithms perform close to MS, but require a big increase in complexity [18, 19, 20, 17].

Evaluating and Comparing Decoders



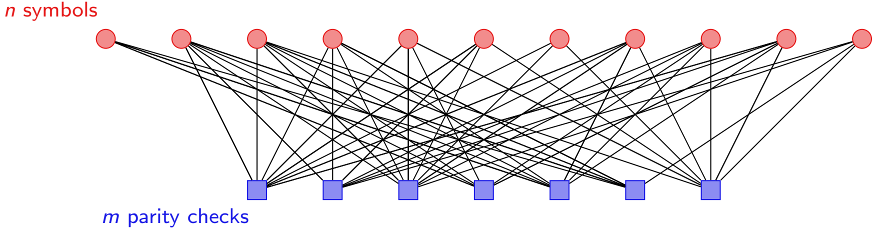
Numerous bit-flipping algorithms have been devised to improve performance. Gradient Descent Bit-Flipping (GDBF) algorithms provide a good balance between performance and complexity [17].

Some bit-flipping algorithms perform close to MS, but require a big increase in complexity [18, 19, 20, 17].

This presentation is about a new GDBF method [1] that offers good performance, without a big increase in complexity.

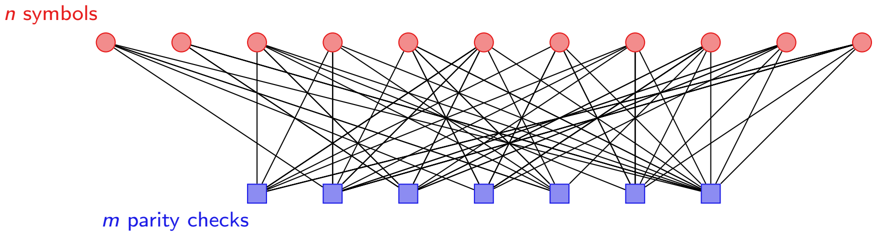
Low-Density Parity-Check Codes

LDPC codes are commonly represented by a Tanner Graph:



Low-Density Parity-Check Codes

LDPC codes are commonly represented by a Tanner Graph:

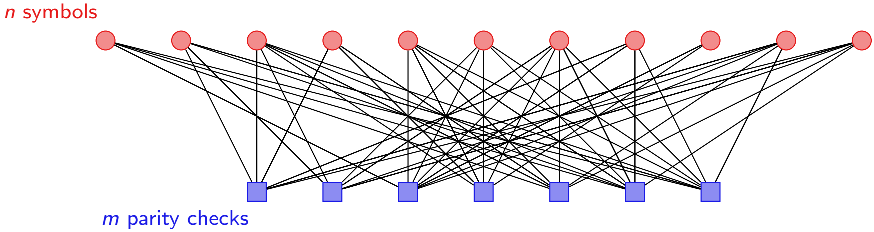


The symbol nodes represent the bits in a codeword.

The parity check nodes represent the **constraints** among the bits.

Low-Density Parity-Check Codes

LDPC codes are commonly represented by a Tanner Graph:



The **edges** indicate **constraint relationships**, i.e.:
If $x_i \in \{-1, +1\}$ are the symbols connected to parity-check node $\mathcal{P}_j$, then they are **constrained** so that

$$s_j = \prod_{i \in N(j)} x_i = +1, \text{ where } N(j) \text{ is the neighborhood of } \mathcal{P}_j.$$

If $s_j = +1$, then parity is satisfied. If $s_j = -1$, then at least one bit has an error.

Bit-Flipping Algorithms

Bit-flipping decoders associate a reliability score to each symbol.

For a given symbol x_i, the reliability score, E_i, represents the sum of all locally available information, including the channel sample magnitude and adjacent parity-check results. If the adjacent parity checks are all good, and the channel confidence is strong, then we shouldn't flip x_i.

For example, suppose:

- $\tilde{y}_i$ is the value received from the channel.
- x_i is the "hypothesis" decision, either $+1$ or -1.
- s_j are the **adjacent** parity-check results ($+1$ is good, -1 is bad).

Bit-Flipping Algorithms

Bit-flipping decoders associate a reliability score to each symbol.

For a given symbol x_i, the reliability score, E_i, represents the sum of all locally available information, including the channel sample magnitude and adjacent parity-check results. If the adjacent parity checks are all good, and the channel confidence is strong, then we shouldn't flip x_i.

For example, suppose:

- $\tilde{y}_i$ is the value received from the channel.
- x_i is the "hypothesis" decision, either $+1$ or -1.
- s_j are the **adjacent** parity-check results ($+1$ is good, -1 is bad).

Then a possible reliability score is:

$$E_i = x_i \tilde{y}_i + \sum_{j \in M(i)} s_j$$

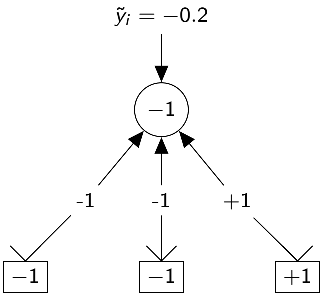
where $M(i)$ is the graph neighborhood of x_i. (This is the score used in GDBF [17])

Single Bit Flipping

For decoding, we can search for the lowest E_i and flip the corresponding x_i.

This is continued until all parity checks are satisfied.

Example: The circle represents x_i



Then $E_i = (-1)(-0.2) + 1 - 1 - 1 = -0.8$.

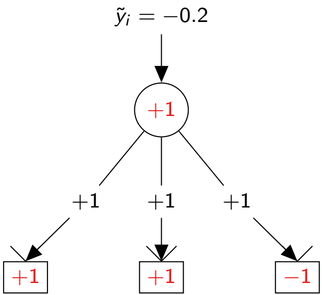
If we flip the bit, then $x_i := +1$.

Single Bit Flipping

For decoding, we can search for the lowest E_i and flip the corresponding x_i.

This is continued until all parity checks are satisfied.

Example: The circle represents x_i



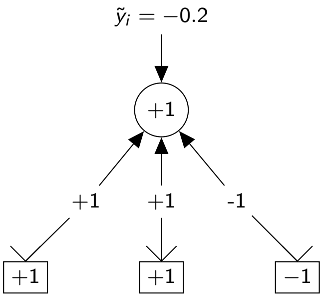
Now we re-evaluate the parity-checks, and they come back as -1, $+1$, and $+1$.

Single Bit Flipping

For decoding, we can search for the lowest E_i and flip the corresponding x_i.

This is continued until all parity checks are satisfied.

Example: The circle represents x_i



Now $E_i = (+1)(-0.2) - 1 + 1 + 1 = 0.8$

In the next iteration, some other bit will be flipped.

Parallel Bit Flipping

Faster decoding is possible by flipping multiple bits each iteration:

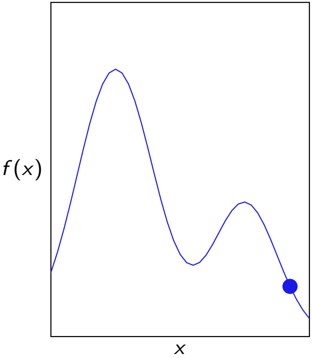
- Set a threshold $\theta < 0$.
- In each iteration, flip all bits for which $E_i < \theta$.

This saves us having to search for the minimum E_i, and allows for fully parallel implementation.

The best θ is found empirically.

Gradient Descent (or Gradient Ascent)

Wadayama showed that bit flipping is related to Gradient Descent Optimization [17].



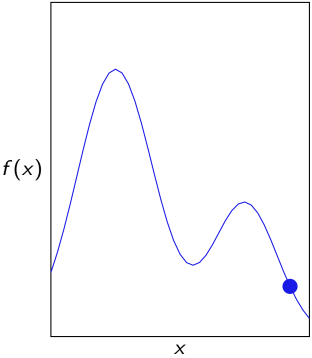
The received samples $\tilde{y}$ provide an initial guess x. This guess is associated with a global reliability metric, called the objective function:

$$f\left(x,\, \tilde{y}\right) = \sum_{i=1}^{n} x_i \tilde{y}_i + \sum_{j=1}^{m} s_j$$

The first part, $\sum_{i=1}^{n} x_i \tilde{y}_i$, represents the standard Maximum Likelihood problem — we want to find the codeword that has highest correlation with the received samples. The second part, $\sum_{j=1}^{m} s_j$, is the sum over all parity checks. If the sequence is valid, then all parity checks equal $+1$.

Gradient Descent (or Gradient Ascent)

Wadayama showed that bit flipping is related to Gradient Descent Optimization [17].

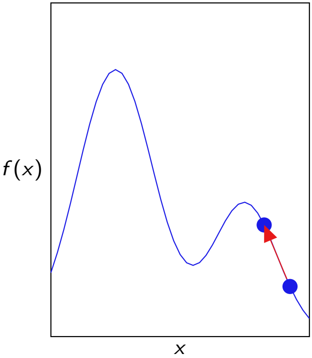


According to the Gradient Descent procedure, we shift the guess toward the objective function gradient:

$$\begin{aligned}\Delta x_i \propto x_i \frac{df}{dx_i} &= x_i \left(\tilde{y}_i + \sum_{j \in M(i)} \prod_{k \in N(j) \setminus i} x_j \right) \\ &= x_i \tilde{y}_i + \sum_{j \in M(i)} s_j \\ &= E_i\end{aligned}$$

Gradient Descent (or Gradient Ascent)

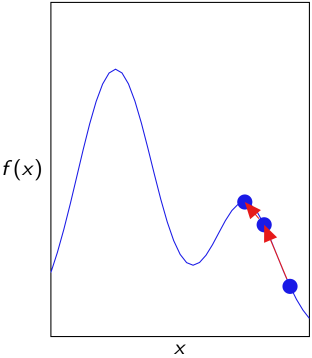
Wadayama showed that bit flipping is related to Gradient Descent Optimization [17].



Bit-flipping incrementally increases the objective function, following the positive slope.

Gradient Descent (or Gradient Ascent)

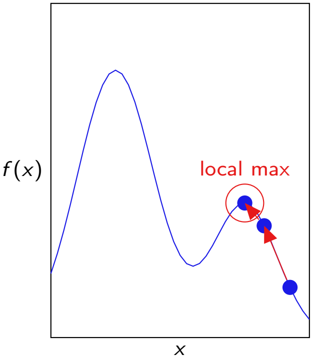
Wadayama showed that bit flipping is related to Gradient Descent Optimization [17].



This procedure tends to get stuck at local maxima.

Gradient Descent (or Gradient Ascent)

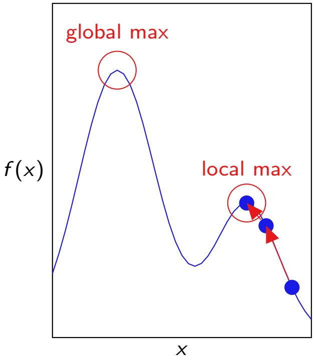
Wadayama showed that bit flipping is related to Gradient Descent Optimization [17].



This procedure tends to get stuck at local maxima.

Gradient Descent (or Gradient Ascent)

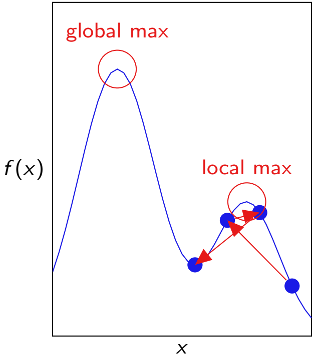
Wadayama showed that bit flipping is related to Gradient Descent Optimization [17].



Several algorithms have been devised to help find the global maximum, but most options add significant complexity.

Stochastic Gradient Descent (or Ascent)

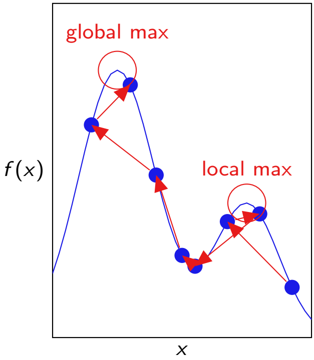
Stochastic Gradient Descent is another well-known optimization heuristic [21, 22, 23, 24].



The guess x gets a random perturbation at each step.

Stochastic Gradient Descent (or Ascent)

Stochastic Gradient Descent is another well-known optimization heuristic [21, 22, 23, 24].

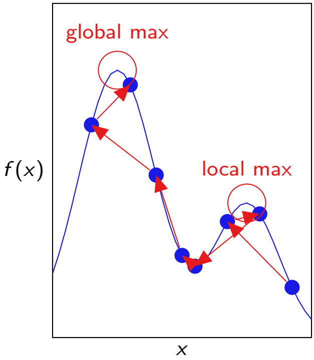


The guess x gets a random perturbation at each step.

The algorithm can randomly escape the local maximum, and is more likely to arrive in the neighborhood of the global maximum.

Stochastic Gradient Descent (or Ascent)

Stochastic Gradient Descent is another well-known optimization heuristic [21, 22, 23, 24].



The guess x gets a random perturbation at each step.

The algorithm can randomly escape the local maximum, and is more likely to arrive in the neighborhood of the global maximum.

In the GDBF algorithm we apply a Gaussian noise perturbation q_i to the reliability metric of every symbol:

$$E_i = x_i \tilde{y}_i + \sum_j s_j + q_i$$

We call this Noisy Gradient Descent Bit-Flipping (NGDBF).

How Much Noise?

Wadayama and others previously tried using a random perturbation to improve bit-flipping performance. They found a very minor improvement [17].

How Much Noise?

Wadayama and others previously tried using a random perturbation to improve bit-flipping performance. They found a very minor improvement [17].

In our method, the perturbations q_i have the same variance as the channel noise. This is much larger than used previously.

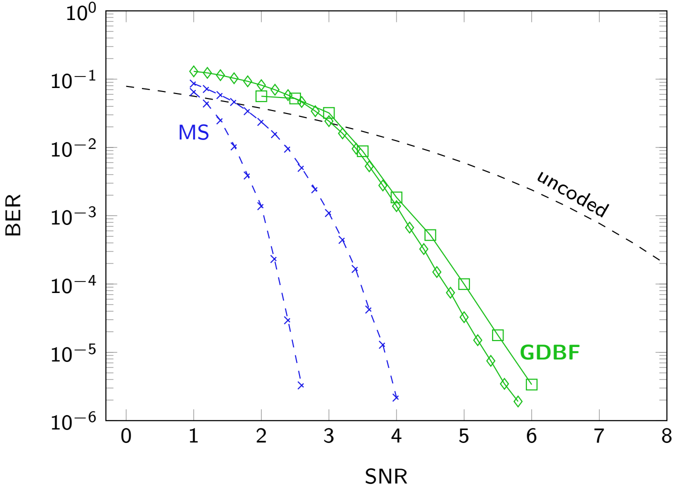
How Much Noise?

Wadayama and others previously tried using a random perturbation to improve bit-flipping performance. They found a very minor improvement [17].

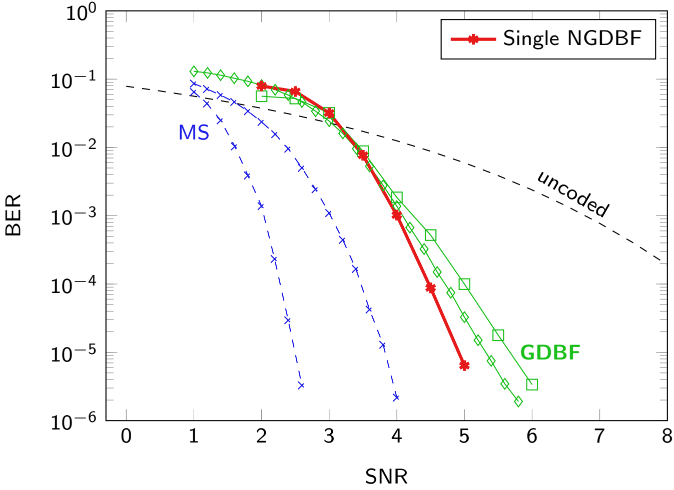
In our method, the perturbations q_i have the same variance as the channel noise. This is much larger than used previously.

Why? We have only intuition to support this approach, but it works...

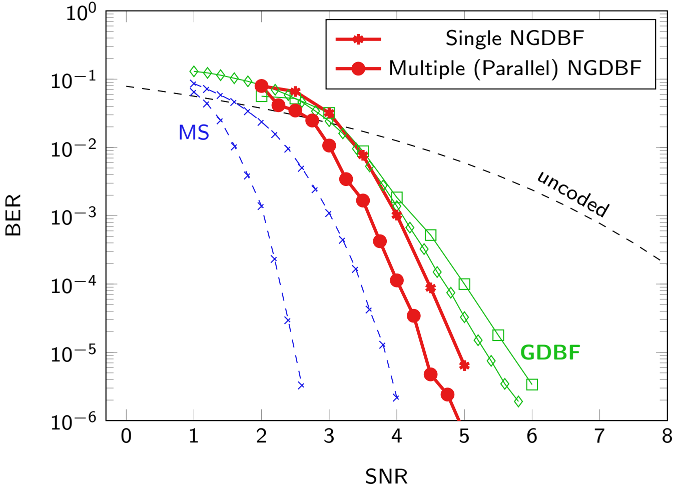
NGDBF Performance



NGDBF Performance

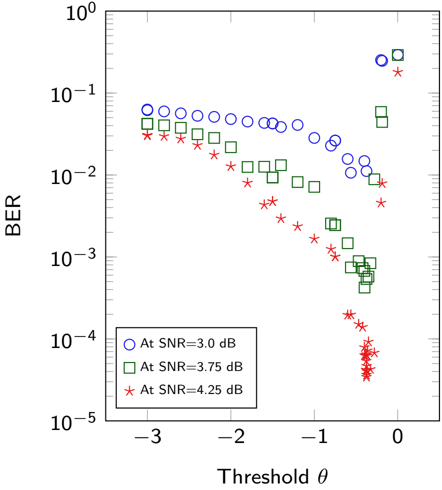


NGDBF Performance



Improvements: Adaptive Thresholds

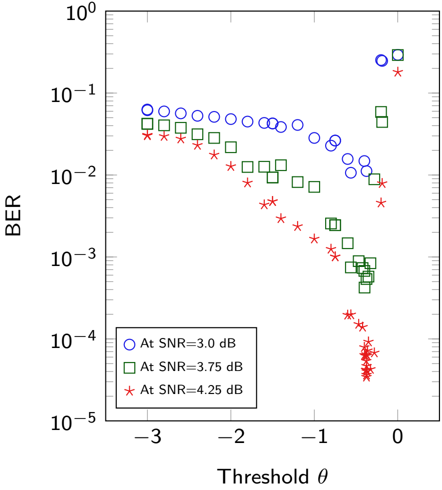
In the parallel bit-flip method, flipping is determined by a threshold θ.



The BER is extremely sensitive to θ.

Improvements: Adaptive Thresholds

In the parallel bit-flip method, flipping is determined by a threshold θ.

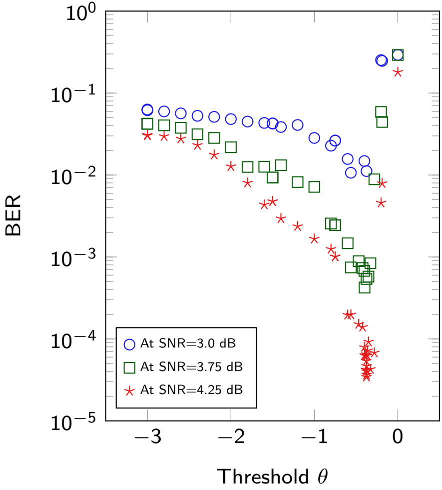


The BER is extremely sensitive to θ.

To reduce sensitivity, we use an adaptive threshold:

Improvements: Adaptive Thresholds

In the parallel bit-flip method, flipping is determined by a threshold θ.



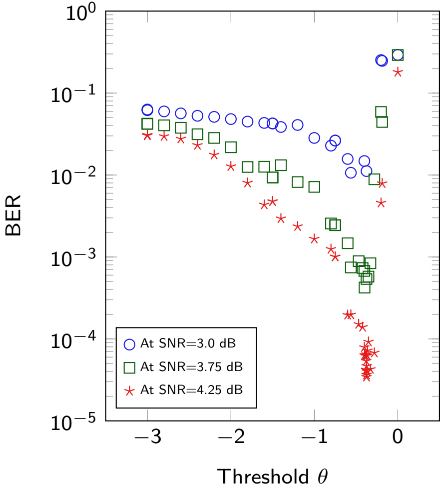
The BER is extremely sensitive to θ.

To reduce sensitivity, we use an adaptive threshold:

Each symbol x_i has a local threshold θ_i.

Improvements: Adaptive Thresholds

In the parallel bit-flip method, flipping is determined by a threshold θ.



The BER is extremely sensitive to θ.

To reduce sensitivity, we use an adaptive threshold:

Each symbol x_i has a local threshold θ_i.

In each iteration, if x_i is flipped, then

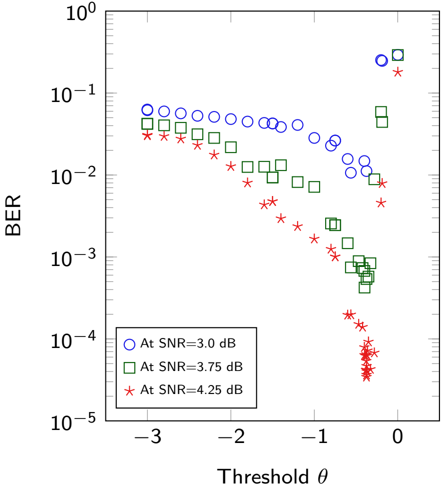
$$\theta_i := \lambda\theta_i$$

If x_i is not flipped, then

$$\theta_i := \lambda^{-1}\theta_i$$

Improvements: Adaptive Thresholds

In the parallel bit-flip method, flipping is determined by a threshold θ.



The BER is extremely sensitive to θ.

To reduce sensitivity, we use an adaptive threshold:

Each symbol x_i has a local threshold θ_i.

In each iteration, if x_i is flipped, then

$$\theta_i := \lambda\theta_i$$

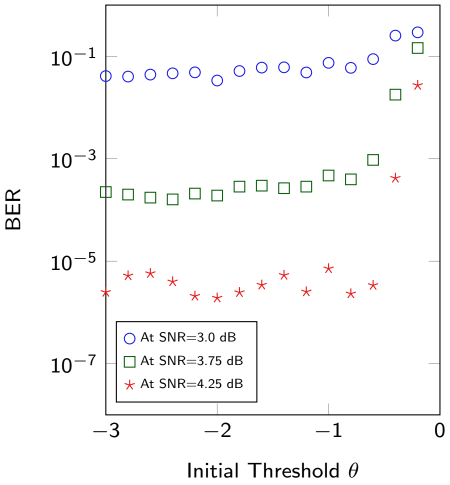
If x_i is not flipped, then

$$\theta_i := \lambda^{-1}\theta_i$$

Typically λ is between 0.90 and 0.99.

Improvements: Adaptive Thresholds

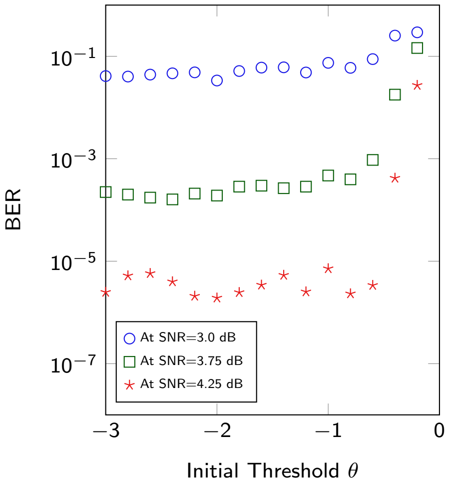
In the parallel bit-flip method, flipping is determined by a threshold θ.



Threshold adaptation reduces parametric sensitivity.

Improvements: Adaptive Thresholds

In the parallel bit-flip method, flipping is determined by a threshold θ.

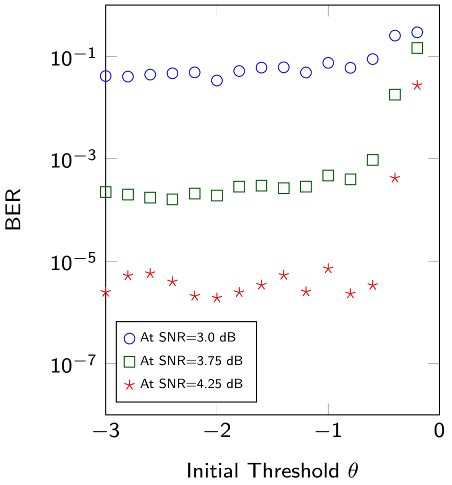


Threshold adaptation reduces parametric sensitivity.

This idea was first proposed by Ismail et al., but did not improve performance (they used it to devise a stopping condition) [25].

Improvements: Adaptive Thresholds

In the parallel bit-flip method, flipping is determined by a threshold θ.

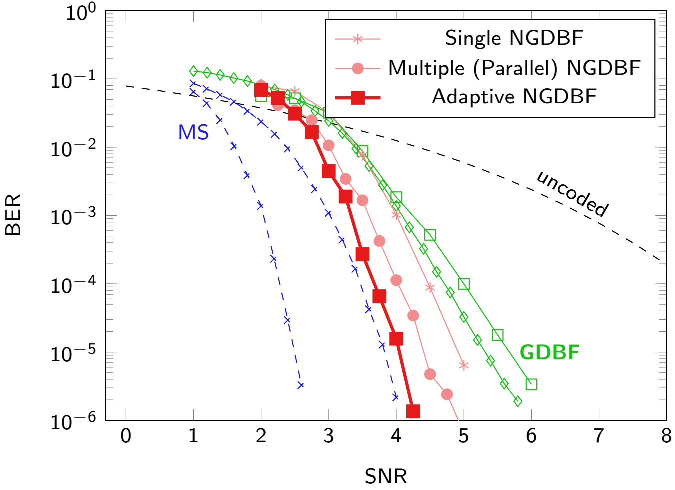


Threshold adaptation reduces parametric sensitivity.

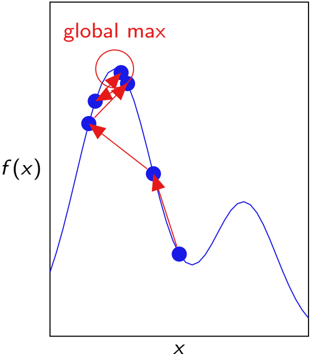
This idea was first proposed by Ismail et al., but did not improve performance (they used it to devise a stopping condition) [25].

Performance is improved by the combination of threshold adaptation with noisy perturbations.

NGDBF Performance with Adaptation



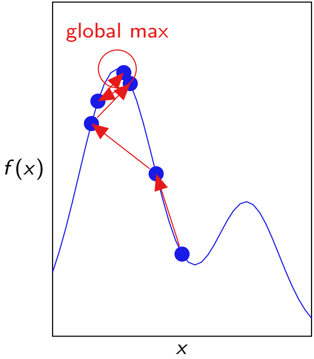
Output Smoothing



Sometimes the noise interferes with convergence.

The state may orbit the solution without reaching it.

Output Smoothing



Sometimes the noise interferes with convergence.

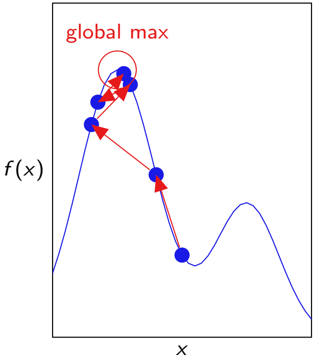
The state may orbit the solution without reaching it.

Performance is improved by smoothing:

- If the guess x hasn't congerged in T iterations,
- Take the decision

$$d_i = \mathrm{sign}\left(\sum_{t=T}^{T+64} x_i\left(t\right)\right)$$

Output Smoothing



Sometimes the noise interferes with convergence.

The state may orbit the solution without reaching it.

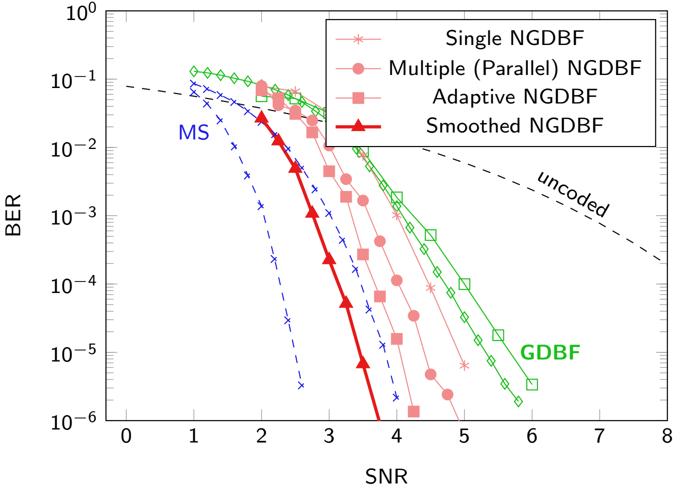
Performance is improved by smoothing:

- If the guess x hasn't congerged in T iterations,
- Take the decision

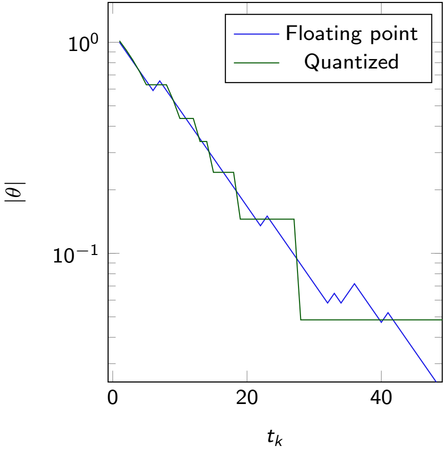
$$d_i = \mathrm{sign}\left(\sum_{t=T}^{T+64} x_i\left(t\right)\right)$$

The implementation is a simple up-down counter.

NGDBF Performance with Adaptation and Smoothing



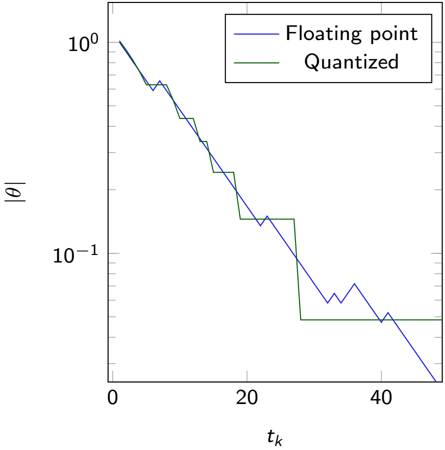
Efficient Implementation



When using quantized samples, the values of θ are also quantized. In this case, only a few distinct θ values can occur.

In this example, with 5-bit quantization only eight $\tilde{\theta}$ values are possible.

Efficient Implementation



We don't need to explicitly multiply by λ or λ^{-1} in each iteration. Instead, we use a counter, t_k, which is incremented whenever x_i is flipped and decremented otherwise. We then select the quantized value of

$$\theta = \theta_0 \lambda^{t_k},$$

which is determined by threshold events in t_k. It is sufficient to simply switch between the quantized $\tilde{\theta}$ values during decoding.

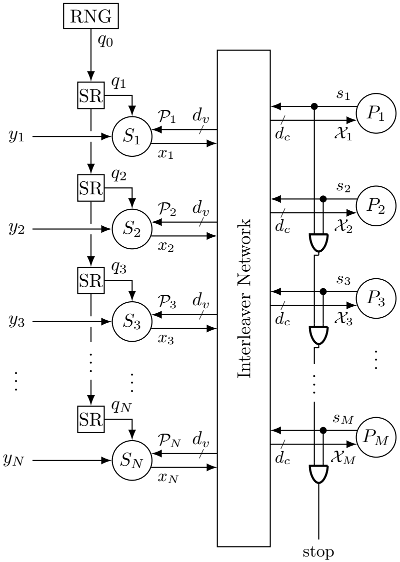
Efficient Implementations

When using quantized arithmetic, the NGDBF modifications have very low complexity:

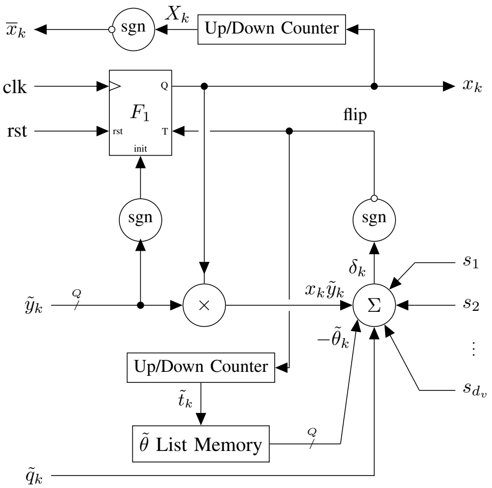
- Smoothing: requires a few toggle flip-flops to implement an up-down counter.
- Threshold adaptation: due to quantization, only a few distinct threshold values are possible.
- Noise samples can be reused without affecting performance.

The end result is only slightly more complex than GDBF.

Decoder Architecture



Symbol Node Architecture



Tradeoffs: Energy, Reliability and Performance

In a communication link, ECC allows reduced transmitter power.

Cost: complex decoding algorithms = increased power in the receiver.

Suboptimal bit-flipping algorithms reduce receiver energy cost.

Big questions:

1. What is the ultimate limit (e.g. threshold) on bit-flipping performance?
2. What is the minimum energy required for decoding?
3. Is there a theoretical relationship between ultimate performance and minimum energy?

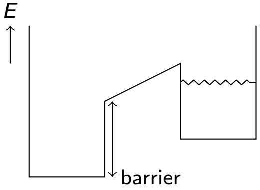
Conventional LDPC Decoders: Minimum Energy

For traditional LDPC algorithms (Belief Propagation and Min-Sum), it is possible to relate performance thresholds with minimum energy-per-bit [2].

We assume a digital architecture, and use Landauer's limit [26] for the minimum energy per switching event:

$$E_{\min} = kT \ln 2$$

Where k is Boltzmann's constant and T is the temperature in K. At room temperature, this evaluates to $E_{\min} = 2.85\,\text{zJ}$ (2.85×10^{-21} J)



Conventional LDPC Decoders: Minimum Energy

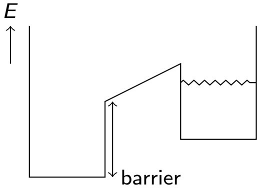
For traditional LDPC algorithms (Belief Propagation and Min-Sum), it is possible to relate performance thresholds with minimum energy-per-bit [2].

We assume a digital architecture, and use Landauer's limit [26] for the minimum energy per switching event:

$$E_{\min} = kT \ln 2$$

Where k is Boltzmann's constant and T is the temperature in K. At room temperature, this evaluates to $E_{\min} = 2.85\,\mathrm{zJ}$ (2.85×10^{-21} J)

Landauer considered a single particle confined to a two-well system. $E_{\min}$ is the minimum work required to move the particle from one well to the other. It is also the minimum barrier height needed to confine the particle.

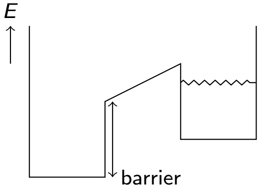


Conventional LDPC Decoders: Minimum Energy

For traditional LDPC algorithms (Belief Propagation and Min-Sum), it is possible to relate performance thresholds with minimum energy-per-bit [2].

When energy approaches the Landauer limit, digital states become unreliable, subject to upsets due to electronic noise, quantum tunneling or other random perturbations [27, 28].

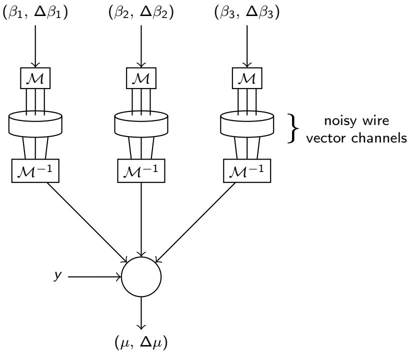
In fact, when the barrier height equals $E_{\min}$, the tunneling probability is 0.5 and there can be no binary state [28]. The practical limit is therefore somewhere higher than $kT \ln 2$.



Conventional LDPC Decoders: Minimum Energy

For traditional LDPC algorithms (Belief Propagation and Min-Sum), it is possible to relate performance thresholds with minimum energy-per-bit [2].

To address the practical limit for LDPC decoders, we account for random upsets by using a modified "density evolution" procedure, which estimates the average switching activity per message while computing the algorithm's performance threshold.

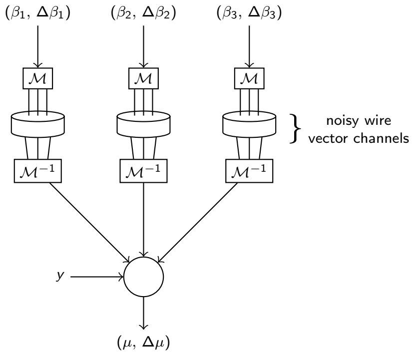


Conventional LDPC Decoders: Minimum Energy

For traditional LDPC algorithms (Belief Propagation and Min-Sum), it is possible to relate performance thresholds with minimum energy-per-bit [2].

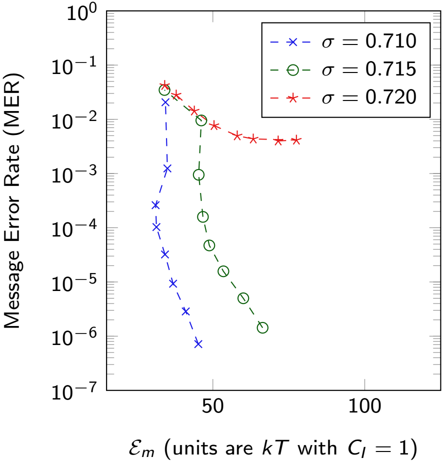
This method assumes a particular digital architecture. Messages are mapped to a physical signal representation via a mapping $\mathcal{M}$, and upsets are randomly inserted into the signals. The upset statistics represent the presence of kT noise, following an approach used by Meindl and Davis[27].

We compute the message statistics at each iteration of the algorithm, jointly tracking the conditional distribution of changes. From these distributions we obtain the switching activity and therefore the limiting energy per bit.



(doesn't work for bit-flipping)

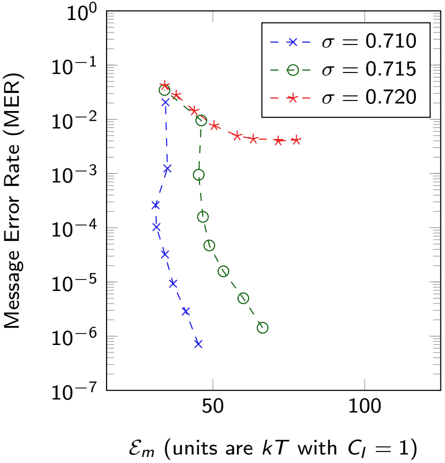
Joint Limit on Performance and Power



By combining switching activity with Landauer's $E_{\min}$ limit, we arrive at a three-way asymptotic relationship:

- Energy-per-Message, $\mathcal{E}_m$ (i.e. power)
- Channel noise parameter σ (related to SNR)
- Decoding threshold (best possible performance)

Joint Limit on Performance and Power

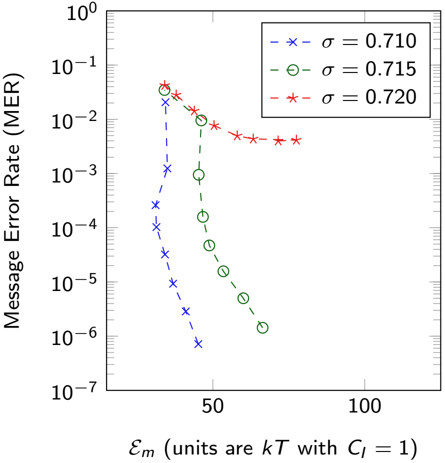


By combining switching activity with Landauer's $E_{\min}$ limit, we arrive at a three-way asymptotic relationship:

- Energy-per-Message, $\mathcal{E}_m$ (i.e. power)
- Channel noise parameter σ (related to SNR)
- Decoding threshold (best possible performance)

For min-sum decoders, we estimate a limiting efficiency of $\approx 10\,\mathrm{aJ}$ per bit (10^{-17} J/bit), which is about four orders of magnitude greater than the Landauer limit for individual switching events.

Joint Limit on Performance and Power



By combining switching activity with Landauer's $E_{\min}$ limit, we arrive at a three-way asymptotic relationship:

- Energy-per-Message, $\mathcal{E}_m$ (i.e. power)
- Channel noise parameter σ (related to SNR)
- Decoding threshold (best possible performance)

For min-sum decoders, we estimate a limiting efficiency of $\approx$ 10 aJ per bit (10^{-17} J/bit), which is about four orders of magnitude greater than the Landauer limit for individual switching events.

These results do not directly apply to bit-flipping algorithms!

Frontier: Noise-Assisted Algorithms

We showed that bit-flipping performance is improved by noise.

Can bit-flipping performance also be improved by random internal upsets?

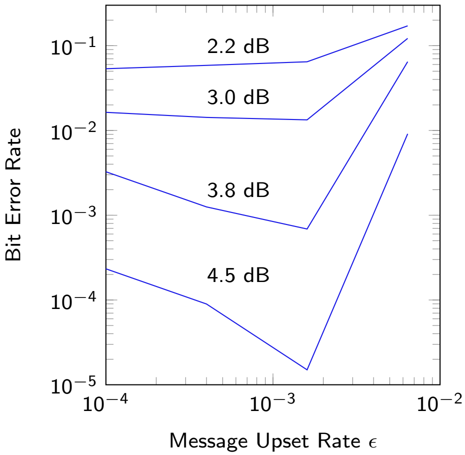
Frontier: Noise-Assisted Algorithms

We showed that bit-flipping performance is improved by noise.

Can bit-flipping performance also be improved by random internal upsets?

Yes!

GDBF with Internal Upsets



We evaluated GDBF performance without the noise terms.

Message upsets were inserted with probability ϵ (an upset means $x_i := -x_i$).

Up to a point, upsets tend to improve the decoder's performance.

This is certainly favorable for operating near the Landauer limit.

Problems for Future Research

Bit-flipping methods rely on heuristic approaches. We need a more complete theory on bit-flipping performance:

- Can we obtain performance thresholds for bit-flipping algorithms?
- Can we develop a better theory of optimality for bit-flipping procedures?
- (Wadayama showed that several BF algorithms can be derived from the gradient descent framework, but gradient descent itself is a family of heuristics.)
- Can we obtain ultimate energy/performance relationships for bit-flipping algorithms? How do they compare to BP and MS?
- Noise-assisted algorithms can get us closer to the Landauer minimum. How much closer?

Acknowledgements

This research was supported by the US National Science Foundation under award ECCS-0954747, and by the Franco-American Fulbright Commission for the international exchange of scholars.

Thank you for listening!

Questions?

References I

Gopalakrishnan Sundararajan, Chris Winstead, and Emmanuel Boutillon. *Noisy Gradient Descent Bit-Flip Decoding for LDPC Codes*. arXiv:1402.2773. 2014. URL: http://arxiv.org/abs/1402.2773.

Chris Winstead and Christian Schlegel. "Energy limits of message-passing error control decoders". In: *Proc. International Zurich Seminar on Communications (IZS)*. 2014. URL: http://left.usu.edu/lab/papers/IZS2014.

Gopalakrishnan Sundararajan Chris Winstead and Emmanuel Boutillon. "A Case Study in Noise Enhanced Computing: Noisy Gradient Descent Bit Flip Decoding". In: *Workshop on Designing With Uncertainty: Opportunities and Challenges*. York, UK, 2014. URL: http://left.usu.edu/lab/?q=node/40.

C. Berrou, A. Glavieux, and P. Thitimajshima. "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1". In: *IEEE Proc. Intern. Conf. on Communications*. Vol. 2. 1993, pp. 1064 –1070. DOI: 10.1109/ICC.1993.397441.

C. Berrou and A. Glavieux. "Near optimum error correcting coding and decoding: Turbo-codes". In: *IEEE Trans. on Communications* 44.10 (1996), pp. 1261 –1271. ISSN: 0090-6778. DOI: 10.1109/26.539767.

References II

- R. Pyndiah. "Near Optiumum Decoding of Product Codes: Block Turbo Codes". In: 42.8 (Aug. 1998).
- Robert G. Gallager. *Low-Density Parity-Check Codes*. 1963.
- D.J.C. MacKay and R.M. Neal. "Near Shannon limit performance of low density parity check codes". In: *IEEE Electronics Lett.* 33.6 (1997), pp. 457 –458. ISSN: 0013-5194. DOI: 10.1049/el:19970362.
- T. Richardson and R. Urbanke. "The renaissance of Gallager's low-density parity-check codes". In: *Communications Magazine, IEEE* 41.8 (2003), pp. 126–131. ISSN: 0163-6804. DOI: 10.1109/MCOM.2003.1222728.
- Sae-Young Chung, T.J. Richardson, and R.L. Urbanke. "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation". In: *Information Theory, IEEE Transactions on* 47.2 (2001), pp. 657–670. ISSN: 0018-9448. DOI: 10.1109/18.910580.
- T.J. Richardson, M.A. Shokrollahi, and R.L. Urbanke. "Design of capacity-approaching irregular low-density parity-check codes". In: *Information Theory, IEEE Transactions on* 47.2 (2001), pp. 619–637. ISSN: 0018-9448. DOI: 10.1109/18.910578.

References III

T.J. Richardson and R.L. Urbanke. “The capacity of low-density parity-check codes under message-passing decoding”. In: *Information Theory, IEEE Transactions on* 47.2 (2001), pp. 599–618. ISSN: 0018-9448. DOI: 10.1109/18.910577.

Niclas Wiberg, Hans-Andrea Loeliger, and Ralf Kotter. “Codes and iterative decoding on general graphs”. In: *European Transactions on Telecommunications* 6.5 (1995), pp. 513–525. ISSN: 1541-8251. DOI: 10.1002/ett.4460060507. URL: http://dx.doi.org/10.1002/ett.4460060507.

N. Wiberg. “Codes and Decoding on General Graphs”. PhD thesis. Linkoping, Sweden: Linkoping University, 1996.

Yu Kou, Shu Lin, and M.P.C. Fossorier. “Low density parity check codes based on finite geometries: a rediscovery”. In: *Information Theory, 2000. Proceedings. IEEE International Symposium on*. 2000, p. 200. DOI: 10.1109/ISIT.2000.866498.

Yu Kou, Shu Lin, and M.P.C. Fossorier. “Low-density parity-check codes based on finite geometries: a rediscovery and new results”. In: *Information Theory, IEEE Transactions on* 47.7 (2001), pp. 2711–2736. ISSN: 0018-9448. DOI: 10.1109/18.959255.

References IV

T. Wadayama et al. “Gradient descent bit flipping algorithms for decoding LDPC codes”. In: *Communications, IEEE Transactions on* 58.6 (2010), pp. 1610–1614. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2010.06.090046.

R. Haga and S. Usami. “Multi-bit flip type gradient descent bit flipping decoding using no thresholds”. In: *Information Theory and its Applications (ISITA), 2012 International Symposium on*. 2012, pp. 6–10.

T. Phromsa-ard et al. “Improved Gradient Descent Bit Flipping algorithms for LDPC decoding”. In: *Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on*. 2012, pp. 324–328. DOI: 10.1109/DICTAP.2012.6215420.

T. Wadayama et al. “Gradient descent bit flipping algorithms for decoding LDPC codes”. In: *Information Theory and Its Applications, 2008. ISITA 2008. International Symposium on*. 2008, pp. 1 –6. DOI: 10.1109/ISITA.2008.4895387.

S. Chakrabartty, R.K. Shaga, and K. Aono. “Noise-Shaping Gradient Descent-Based Online Adaptation Algorithms for Digital Calibration of Analog Circuits”. In: *Neural Networks and Learning Systems, IEEE Transactions on* 24.4 (2013), pp. 554–565. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2012.2236572.

References V

William A Gardner. "Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique". In: *Signal Processing* 6.2 (1984), pp. 113–133.

Nicolas Meuleau and Marco Dorigo. "Ant colony optimization and stochastic gradient descent". In: *Artificial Life* 8.2 (2002), pp. 103–121.

Tong Zhang. "Solving large scale linear prediction problems using stochastic gradient descent algorithms". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 116.

M. Ismail et al. "Low latency low power bit flipping algorithms for LDPC decoding". In: *Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on*. 2010, pp. 278 –282. DOI: 10.1109/PIMRC.2010.5671820.

Rolf Landauer. "Irreversibility and heat generation in the computing process". In: *IBM journal of research and development* 5.3 (1961), pp. 183–191.

James D Meindl and Jeffrey A Davis. "The fundamental limit on binary switching energy for terascale integration (TSI)". In: *Solid-State Circuits, IEEE Journal of* 35.10 (2000), pp. 1515–1516.

Victor V Zhirnov et al. "Limits to binary logic switch scaling-a gedanken model". In: *Proceedings of the IEEE* 91.11 (2003), pp. 1934–1939.