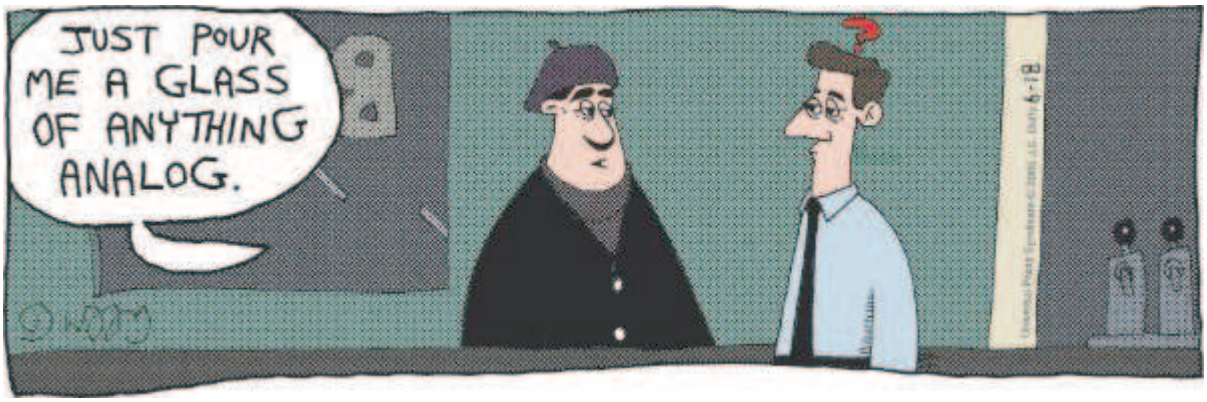
Iterative Decoding via Analog Processing

Christian Schlegel

High-Capacity Digital Communications (HCDC) Lab
Department of Electrical Engineering
University of Alberta
Edmonton, AB, CANADA
Email: schlegel@ece.ualberta.ca
Website: www.ece.ualberta.ca/hcdc

Italian Summer School

Seminar Notes, June 2005



Rate, Power, and Complexity

It all started with Shannon in 1948

Shannon's capacity formula (1948) for the additive white Gaussian noise channel (AWGN):

$$C = W \log_2 (1 + S/N) \ \ [\text{bits/second}]$$

- W is the bandwidth of the channel in Hz
- S is the signal power in watts
- N is the total noise power of the channel watts

Channel Coding Theorem (CCT):
The theorem has two parts.

1. Its **direct part** says that for rate $R < C$ there exists a coding system with arbitrarily low block and bit error rates as we let the codelength $N \to \infty$.
2. The **converse part** states that for $R \geq C$ the bit and block error rates are strictly bounded away from zero for any coding system

The CCT therefore establishes rigid limits on the maximal supportable transmission rate of an AWGN channel in terms of power and bandwidth.

Normalized Capacity

For finite-dimensional channels the following discrete capacities hold:

$$C_d = \tfrac{1}{2}\log_2\left(1 + 2\tfrac{R_d E_b}{N_0}\right) \ \text{[bits/dimension]}$$

$$C_c = \log_2\left(1 + \frac{RE_b}{N_0}\right) \ \text{[bits/complex dimension]}$$

There are a maximum of approximately 2.4 dimensions per unit Bandwidth and Time

The *Shannon bound* per dimension is given by

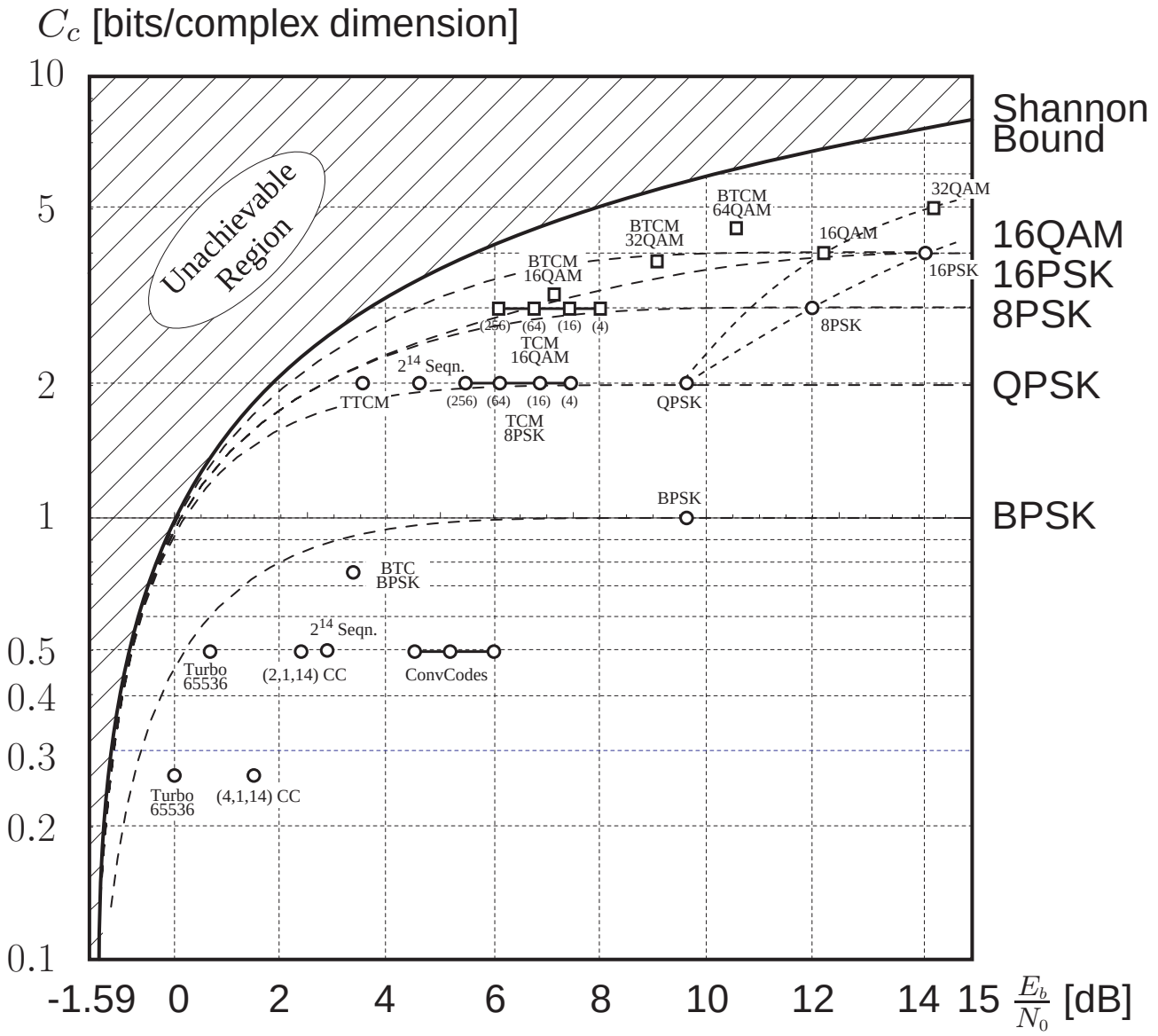
$$\frac{E_b}{N_0} \geq \frac{2^{2C_d} - 1}{2C_d}; \qquad \frac{E_b}{N_0} \geq \frac{2^{C_c} - 1}{C_c}.$$

System Performance Measure In order to compare different communications systems, we need a parameter expressing the performance level. It is the information bit error probability P_b and typically falls into the range $10^{-3} \geq P_b \geq 10^{-6}$.

[WoJ65] J.M. Wozencraft and I.M. Jacobs, *Principles of Communication Engineering*, John Wiley & Sons, Inc., New York, 1965, reprinted by Waveland Press, 1993.

Examples:

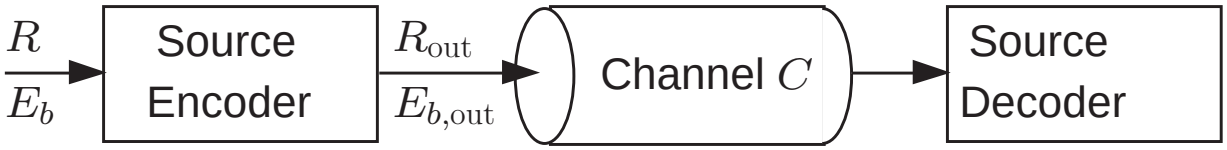
Spectral Efficiencies versus power efficiencies of coded and uncoded digital transmission systems at a **bit error rate** of $P_b = 10^{-5}$:



[SchPer04] C. Schlegel and L. Perez, *Trellis and Turbo Coding*, IEEE Press, Piscataway, NJ, 2004.

Finite Error Probabilities

If we are willing to accept a non-zero finite error rate P_b on the decoded bits, the available resources can be stretched.



Lossy Source Compression can achieve a compression from rate $R \to R_{\text{out}}$ if we accept a reconstruction error probability of P_b. Then

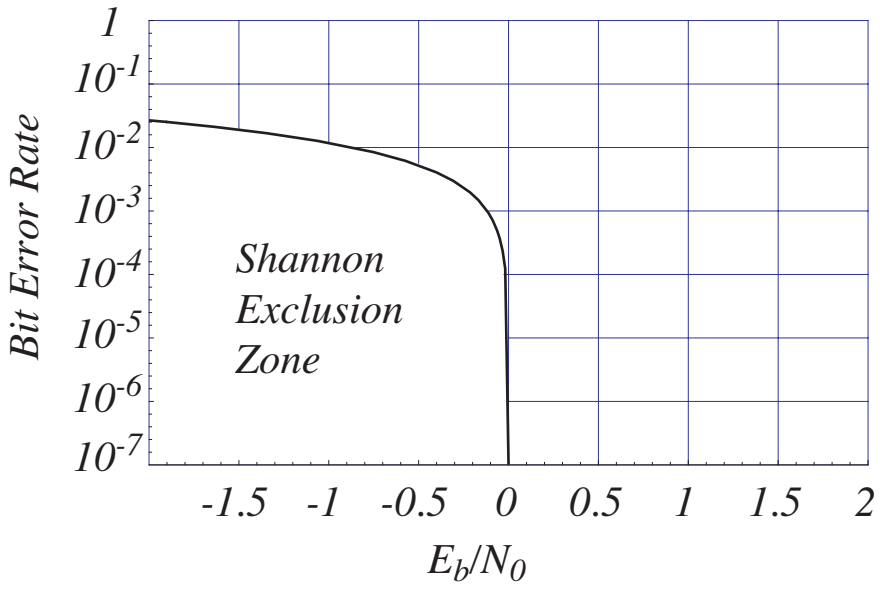
$$R_{\text{out}} = (1 - h(P_b))R$$

binary entropy function: $h(p) = -p\log_{10}(p) - (1-p)\log_{10}(1-p)$

The rate now has to obey: $R_{\text{out}} \leq C$

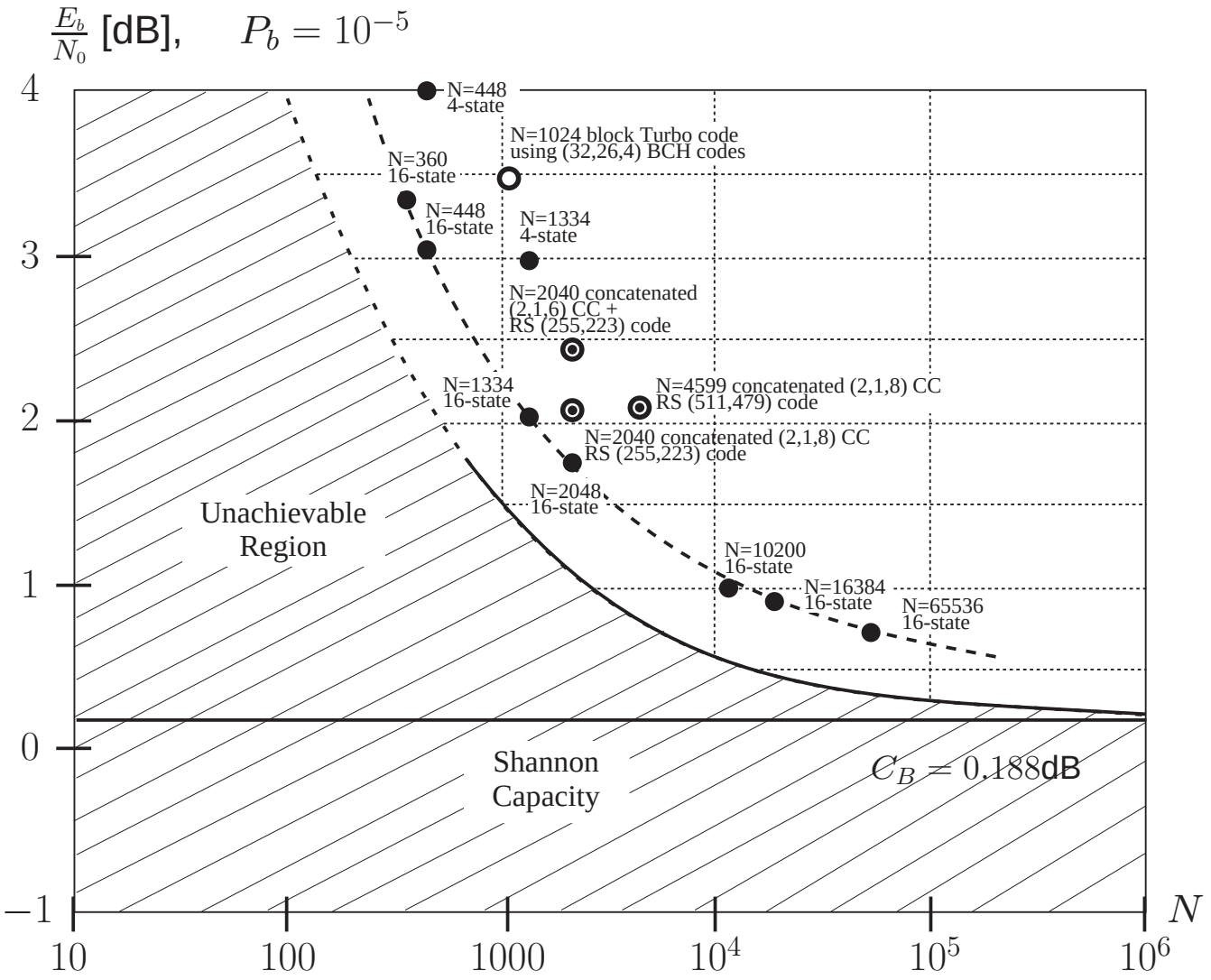
which leads to the modified Shannon bound:

$$\frac{E_b}{N_0} \geq \frac{2^{(1-h(P_b)\eta} - 1}{\eta}(1 - h(P_b))$$



Code Efficiency

codes perform better if they are larger. Here plotted for $R = 0.5$.



[SGB67] C.E. Shannon, R.G. Gallager, and E.R. Berlekamp, "Lower bounds to error probabilities for coding on discrete memoryless channels," *Inform. Contr.,* vol. 10, pt. I, pp. 65–103, 1967, Also, *Inform. Contr.,* vol. 10, pt. II, pp. 522-552, 1967.

[ScP99] C. Schlegel and L.C. Perez, "On error bounds and turbo codes,", *IEEE Communications Letters*, Vol. 3, No. 7, July 1999.

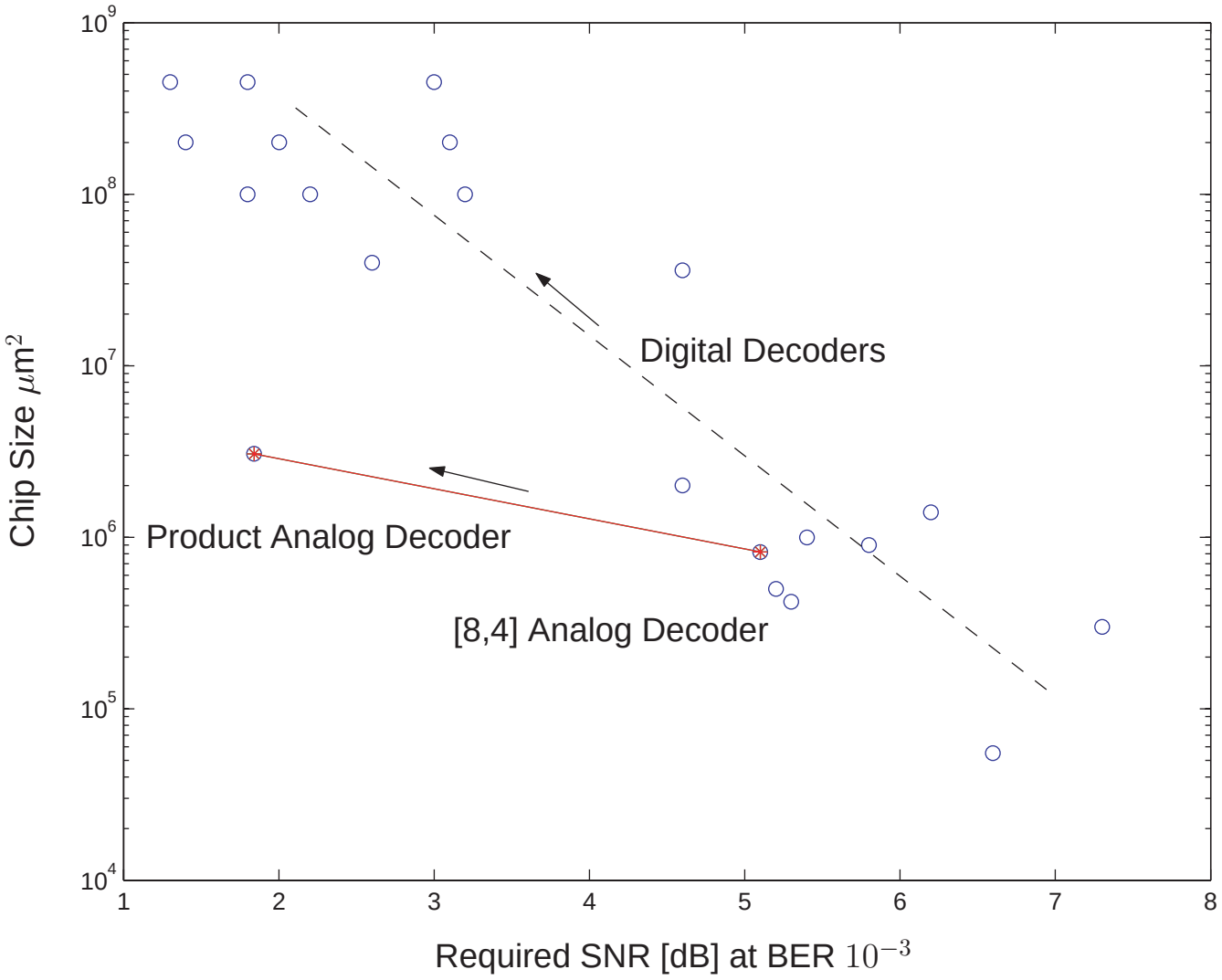
System Complexity: Real-World Issues

Apart from the **Algorithmic Computational Complexity**, the following complexity measures are important for implementations:

- Size of a VLSI Implementation
- Power Dissipation per Decoded Bit
- Implementation and Verification Complexity

Digital Decoder Implementations require a VLSI implementation size which empirically follows an inverse power law of the the required SNR.

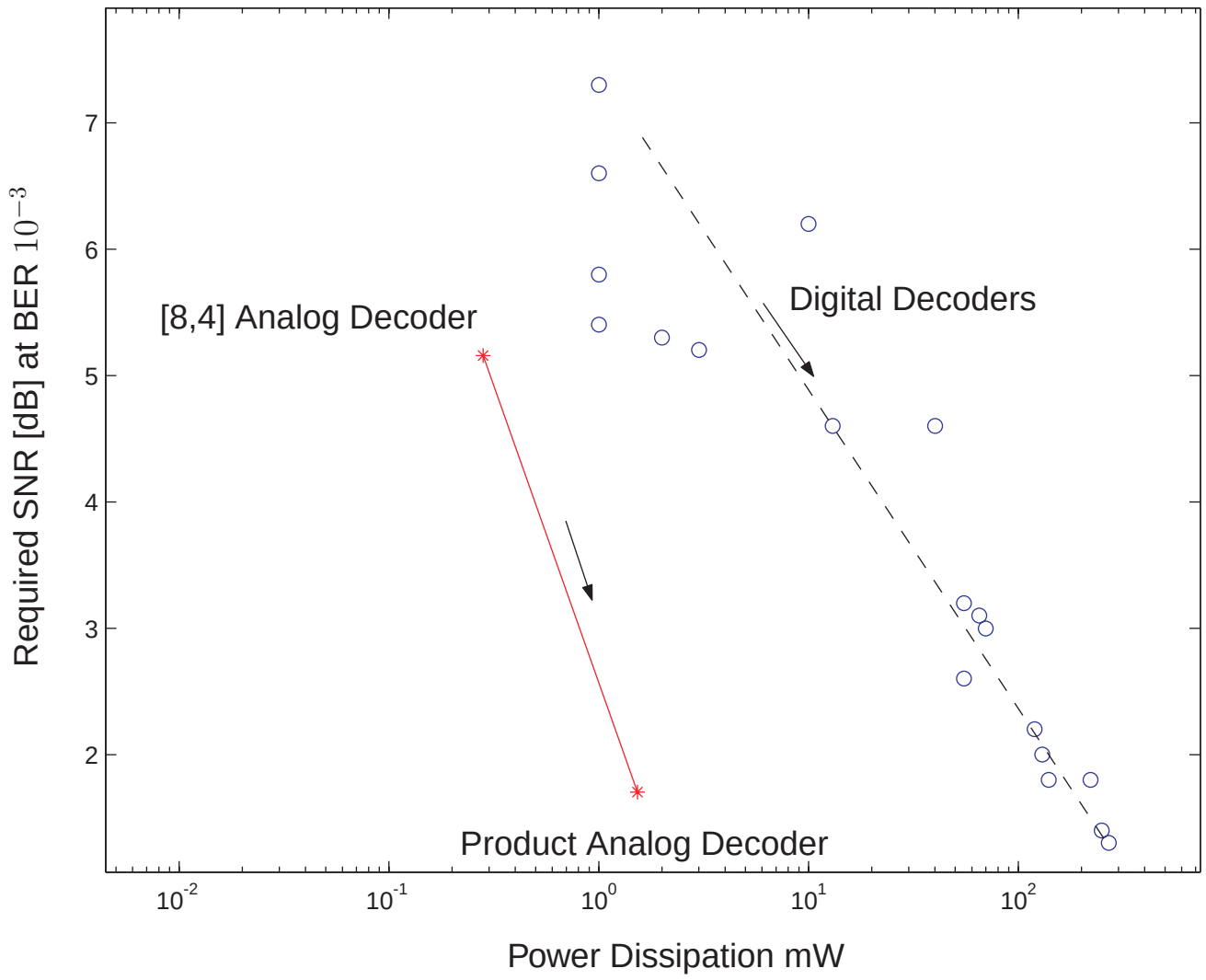
Analog Decoder Implementations: appear to have a substantial size advantage:



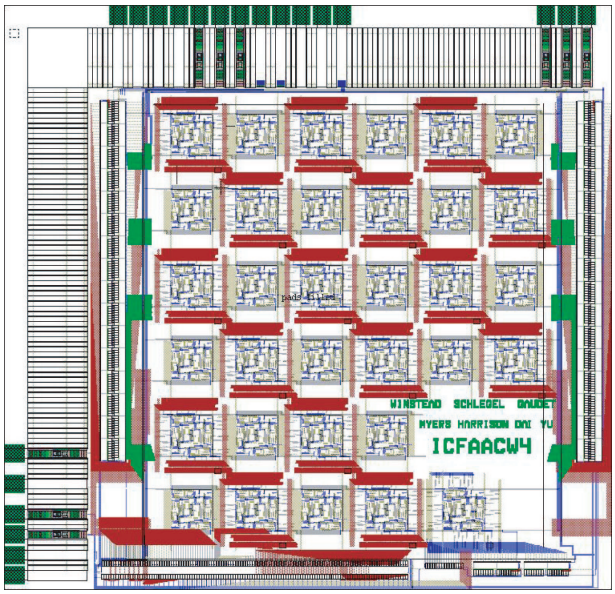
Power Dissipation

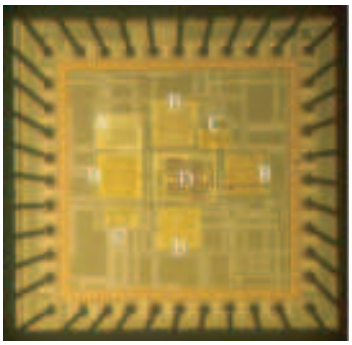
The power dissipated per decoded bit is an important measure of decoder complexity. No coherent theory is known at this point. It also seems to follow as a power function the required signal-to-noise ratio.

Analog Decoder Implementations: appear to have an even stronger substantial advantage in the decoding power dissipation:



Experimental Chips





Analog Decoders have the potential to be extremely power efficient:

Code	Proc.	Power	Speed	Energy/Bit
small turbo	0.35μ	185mW 3.3V	13.3 Mb	13.9nJ/b
(8,4) Hamming	0.5 μ	45mW 3.3V	1 Mb	45nJ/b
1024 LDPC	0.16μ	690mW 1.5V	500 Mb	1.26nJ/b
convolutional	0.25μ	20mW 3.3V	160 Mb	0.125nJ/b

U of A Chips

$(16,11)^2$ product	0.18μ	7mW 1.8V	100 Mb v	0.07nJ/b
(8,4) Low voltage	0.18μ	36μW 1.8V	4.4 Mb	0.008nJ/b
(8,4) Low voltage	0.18μ	150μW 0.8V	3.7Mb	0.042nJ/b
(8,4) Low voltage	0.18μ	2.4μW 0.5V	69kb	0.034nJ/b

Comments:

- Numbers in red are actual measurements of test chips.
- Measurements include IO power and interface losses.
- Brain uses an estimated 10pJ/processed bit

Turbo Codes

Claude Berrou's Turbo Codes have

- opened a new (and final) chapter in error control coding
- opened the flood gates for iterative decoding and iterative signal processing [SchGra05].
- have motivated the novel field of analog processing of digital data.

The author and Claude Berrou enjoying a cigar

[Guiz04]	E. Guizzo, "Closing in on the Perfect Code," *IEEE Spectrum*, Vol. 41, No. 3, March 2004, pp. 36–42.
[SchGra05]	. Schlegel and A. Grant, *Coordinated Multiple User Communications*, Springer Publishers, 2005.

Low-Density Parity-Check Codes

Low-Density Parity-Check Codes

- Low Density Parity Check (LDPC) codes where introduced in the dissertation of Robert G. Gallager in 1960 [Gall62, Gall63].
- Like Turbo Codes, LDPC are decoded with an iterative algorithm based on message passing.
- LDPC codes are now enjoying a renaissance and are considered an attractive alternative to parallel concatenated convolutional codes for near capacity performance.

[Gall62]	R. G. Gallager, "Low-density parity-check codes", *IRE Trans. on Inform. Theory*, pp. 21–28, Vol. 8, No. 1, January 1962.
[Gall63]	R.G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, MA,1963.
[Mac99]	D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices", *IEEE Trans. Inform. Theory*, vol IT-45, No. 2, pp. 399–431, March, 1999.

Linear Block Codes: Some Background

- A binary block encoder maps binary input (source) sequences, **u** of length K to binary codewords, **v**, of length N. The rate of such a code is

$$R = \frac{K}{N}$$

- A rate K/N linear block code can be fully described by a $K \times N$ **generator** matrix **G**. Given **G**, encoding may be accomplished by simple matrix multiplication, i.e.,

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G}$$

- In **systematic** form, the generator matrix takes the form

$$\mathbf{G} = [\mathbf{I}_K | \mathbf{P}] ,$$

  where $\mathbf{I}_K$ is the $K \times K$ identity matrix. In this case, the codeword takes the form

$$\mathbf{v} = (\underbrace{u_0, u_1, \cdots, u_{K-1}}_{K \text{ information bits}}, \underbrace{p_0, p_1, \cdots, p_{n-k-1}}_{N-K \text{ parity bits}})$$

- A linear block code may be described by a $(N-K) \times N$ **parity check** matrix $\mathbf{H}$. An N bit sequence $\mathbf{r}$ is a codeword if and only if

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = \mathbf{0}$$

  The $(N-K)$-tuple $\mathbf{s}$ is called the **syndrome**.

- For systematic codes,

$$\mathbf{H} = \left[\mathbf{I}_{N-K} | \mathbf{P}^T\right] .$$

Gallager Codes

Gallager defined LDPC codes using sparse parity check matrices consisting almost entirely of zeroes.

An (N, p, q) Gallager code of length N specified by a parity check matrix $\mathbf{H}$ with **exactly** p ones per column and **exactly** q ones per row and where $p \geq 3$. The desired code dimension K must also be chosen.

$$
\begin{array}{cccccccccccccccccccc}
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\
\hline
\mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \\
\hline
\mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{1}
\end{array}
$$

Random Construction: The actual $(N - K) \times N$ parity check matrix $\mathbf{H}$ may be constructed randomly subject to these constraints. **Rate:** If all the rows of $\mathbf{H}$ are linearly independent then the code rate is

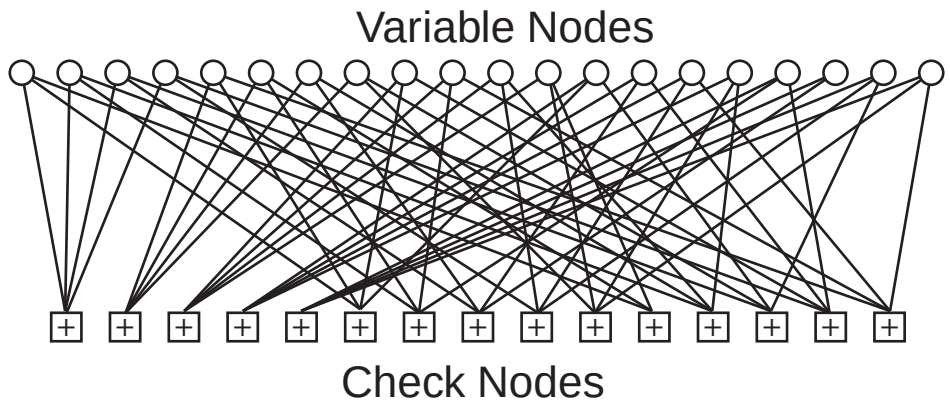
$$R = \frac{N - (N - K)}{N} = 1 - \frac{p}{q}$$

Linear dependence results in **higher** rate codes.

Graphical Code Representation

LDPC codes are preferably represented by a bi-partite graph, where one class of nodes represents the variables (**Variable Nodes**) and the other class represents the (**Check Nodes**):

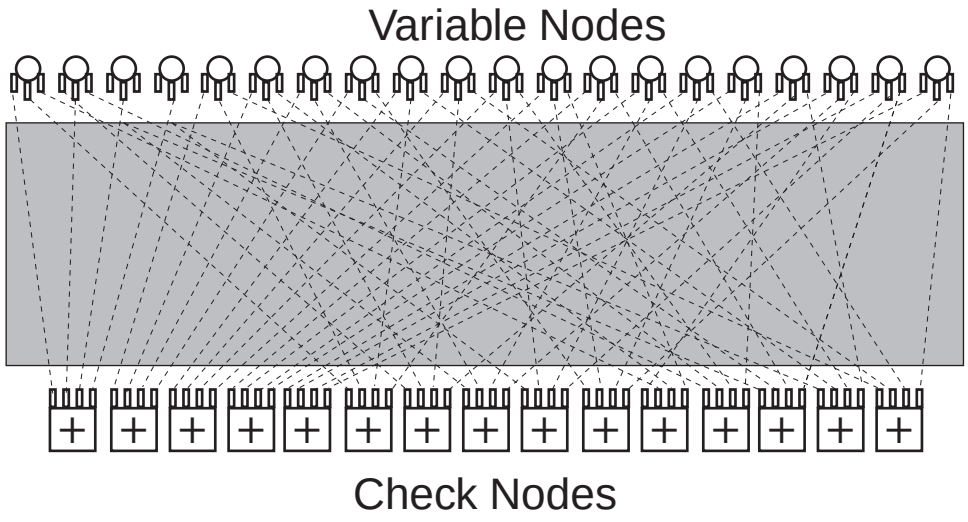
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1



Regular LDPC Codes have a fixed number of branches d_v leaving each variable node, and a fixed number of branches d_c leaving each check node.

Regular LDPC Codes

Code Specification lies in the interconnection network:



Interleaver – Connection Network Each connection point at a node is called a **socket**. There are then $d_v N = d_c(N - K)$ such sockets.

Code Definition

A regular LDPC code is completely defined by a permutation $\pi(i)$ of the natural numbers $1 \leq i \leq d_v N$. The index i refers to the sockets number at the variable nodes, and $\pi(i)$ to the socket number at the check nodes to which socket i connects.

Irregular LDPC Codes

It was observed [Luby01] that **irregular LDPC codes** can provide a performance of up to 0.8dB better for large codes than regular LDPC codes.

Degree Distribution: An irregular code is specified by a degree distribution:

$$\gamma(x) = \sum_i \gamma_i x^{i-1}; \quad \gamma(1) = 1$$

The coefficients γ_i denote the fraction of **edges** which are connected to a node of degree i.

Code Definition (Irregular LDPCs)

An irregular LDPC code is completely defined by a permutation by two degree distributions $\lambda(x)$ for the variable nodes, and $\rho(x)$ for the parity check nodes, together with a permutation $\pi(i)$ of variable socket numbers to check socket i numbers.

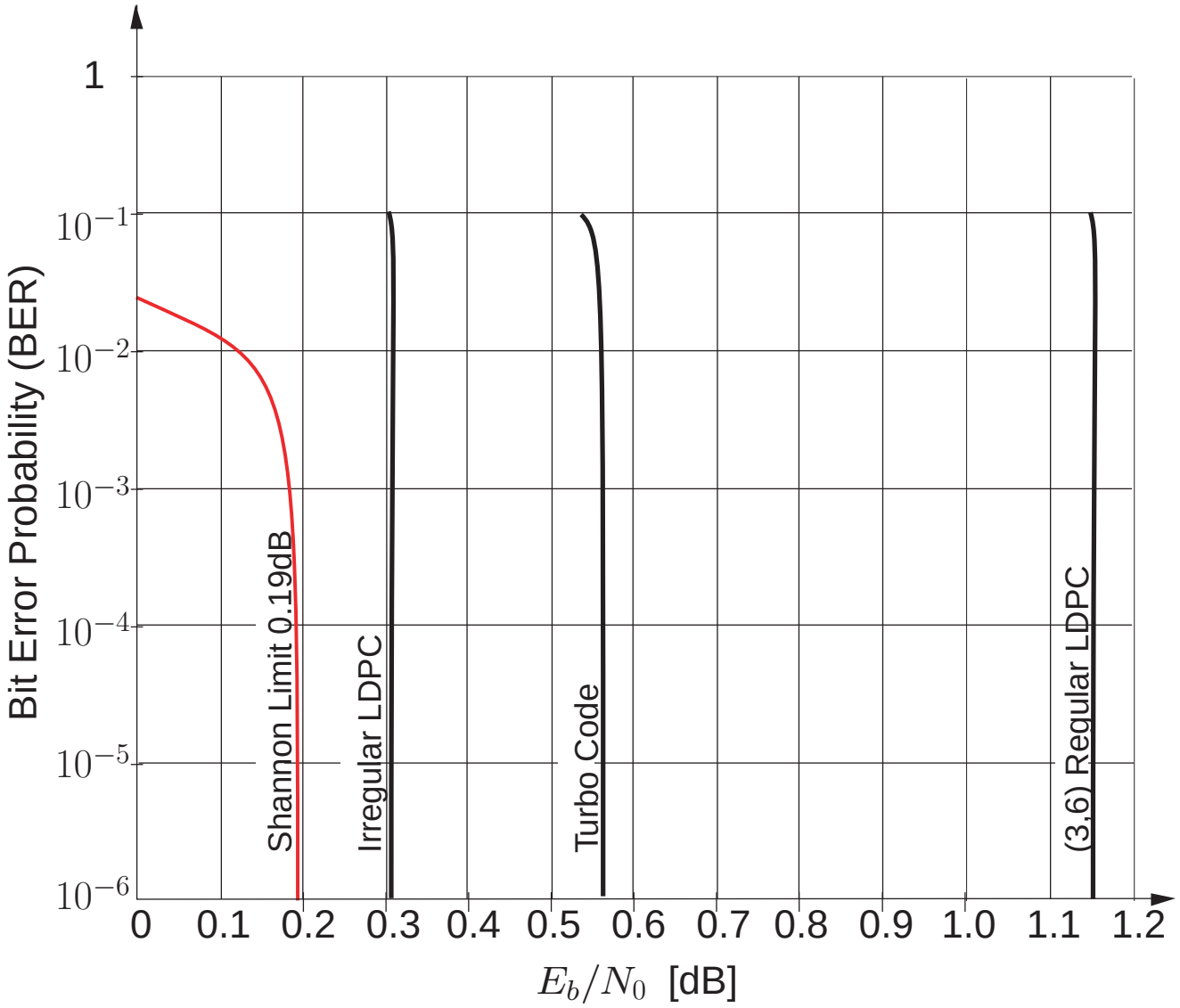
Design Rate of an irregular LDPC code is given as

$$R = 1 - \frac{N-K}{N} = 1 - \frac{\sum_i \frac{\rho_i}{i}}{\sum_i \frac{\lambda_i}{i}}$$

[Luby01] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Improved low-density parity-check codes using irregular graphs", *IEEE Trans. Inform. Theory,* Vol. 47, No. 2, pp. 585–598, 2001.

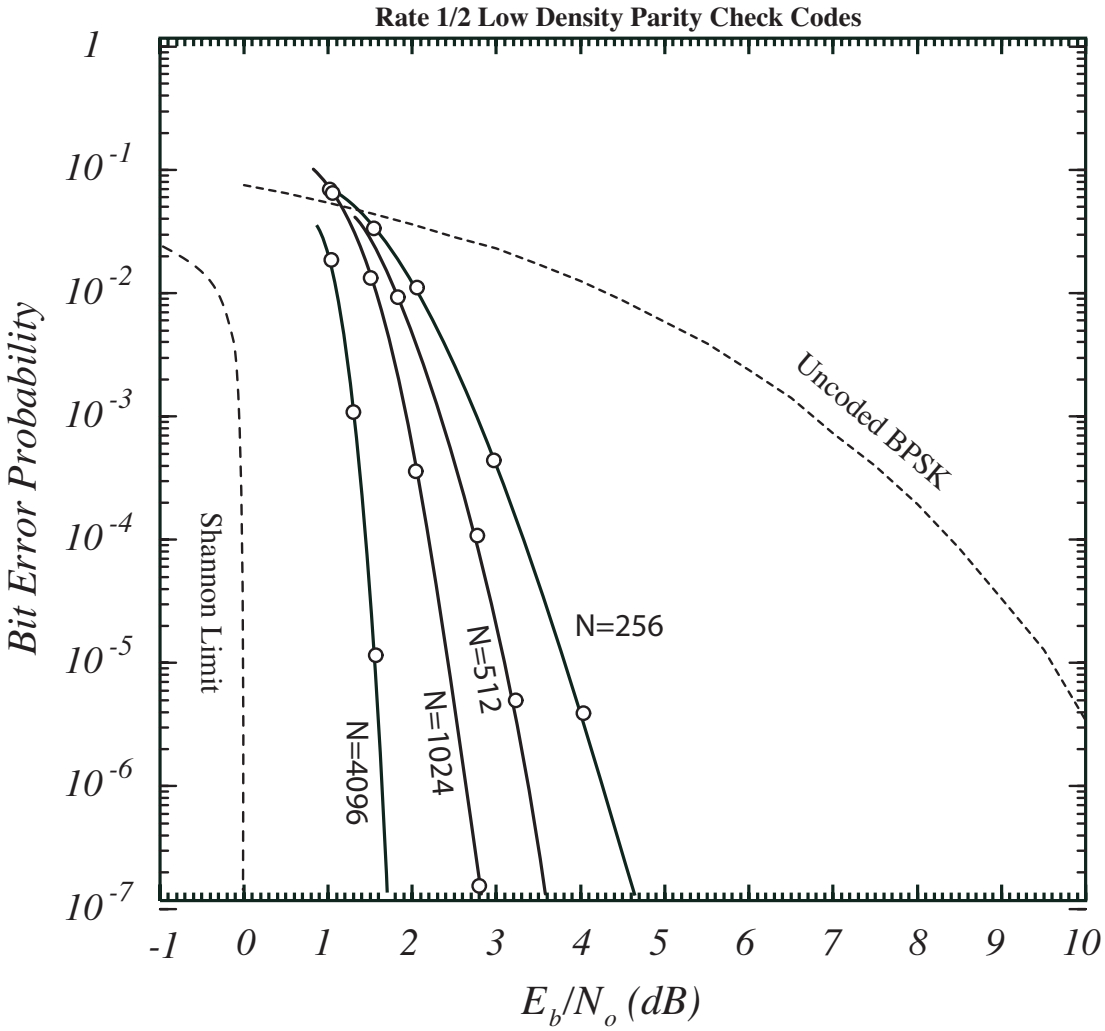
Performance of Gallager Codes: Large Codes

Simulation of Codes of Length 10^6



For competitive performance, irregular LPDC codes are required at low rates.

Performance of Gallager Codes: Finite-Size Codes



- This figure shows the performance of rate $R = 1/2$ Gallager LPDC codes on the AWGN channel with soft decision iterative decoding.

- Irregular codes offer better performance than regular codes, sometimes up to 0.8dB!

Message Passing Decoding: AWGN Channels

Step 1: Initialize $\lambda_i = \frac{2}{\sigma^2} r_i$ for each variable node.

Step 2: Variable nodes send $\mu_{i \to j} = \lambda_i$ to each check node $j \in V_i$.

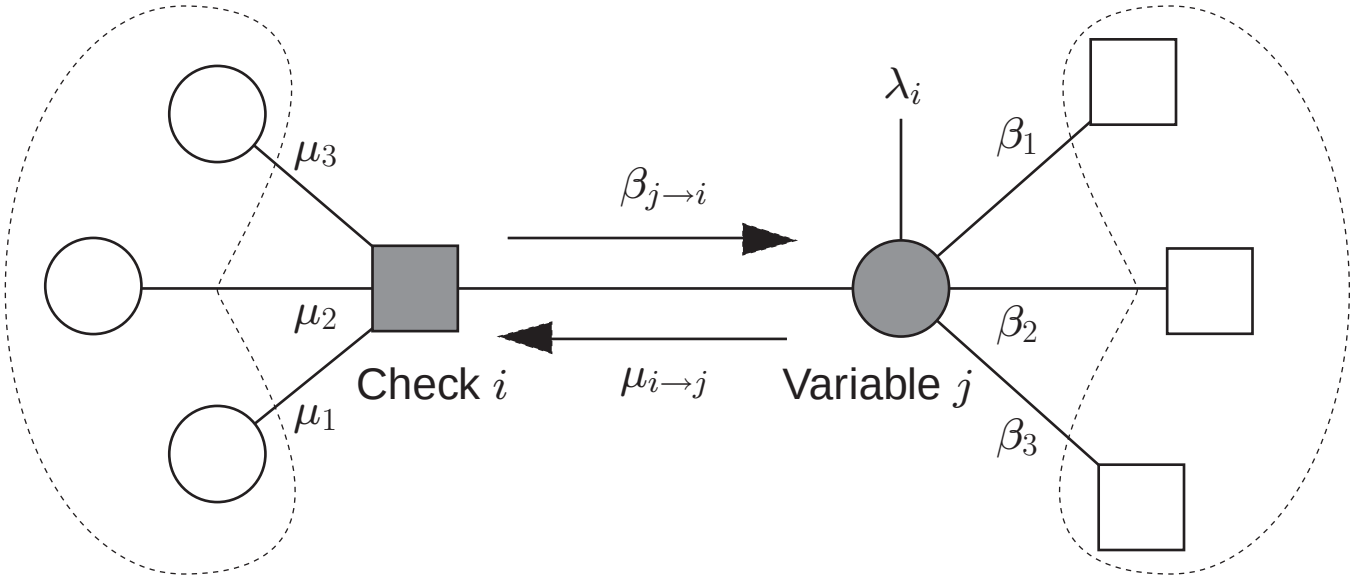
Step 3: Check nodes connected to variable node i send

$$\beta_{j \to i} = 2 \tanh^{-1} \left(\prod_{i \in C_{i \setminus j}} \tanh \left(\frac{\mu i \to j}{2} \right) \right),$$

Step 4: Variable nodes connected to check nodes j send

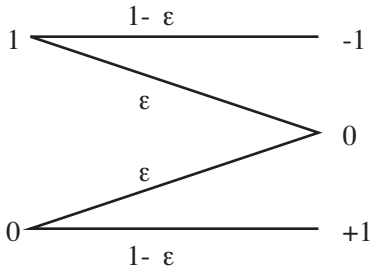
$$\mu_{i \to j} = \sum_{j \in V_{i \setminus j}} \beta_{j \to i} + \lambda_i$$

Step 5: When a fixed number of iterations have been completed or the estimated codeword $\hat{\mathbf{x}}$ satisfies the syndrome constraint $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$ stop. Otherwise return to Step 3.



The Binary Erasure Channel (BEC)

The erasure channel is a simple test example for coding ideas:



Decoding on the BEC follows the following simple algorithm:

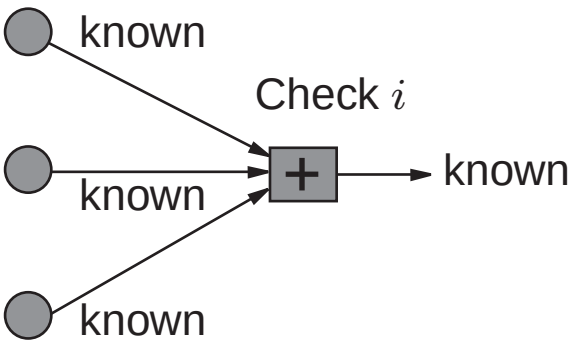
Step 1: Initialize $d_i = r_i$ for each variable node. If $r_i = 0$ then the received symbol i has been erased and variable i is *unknown*.

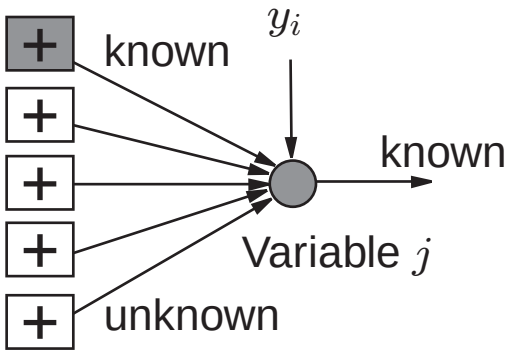
Step 2: Variable nodes send $\mu_{i\to j} = d_i$ to each check node $j \in V_i$.

Step 3: Check nodes connected to variable node i send $\beta_{j\to i} = \prod_{l\in C_{j\setminus i}} \mu_{l\to j}$ to i. That is, if all incoming messages are different from zero, the check node sends back to i the value that makes the check consistent, otherwise it sends back a zero for "unknown".

Step 4: If the variable i is unknown, and at least one $\beta_{j\to i} \neq 0$, set $d_i = \beta_{j\to i}$ and declare variable i to be *known*.

Step 5: When all variables are known, or after a pre-described number of iterations, stop. Otherwise go back to Step 2.



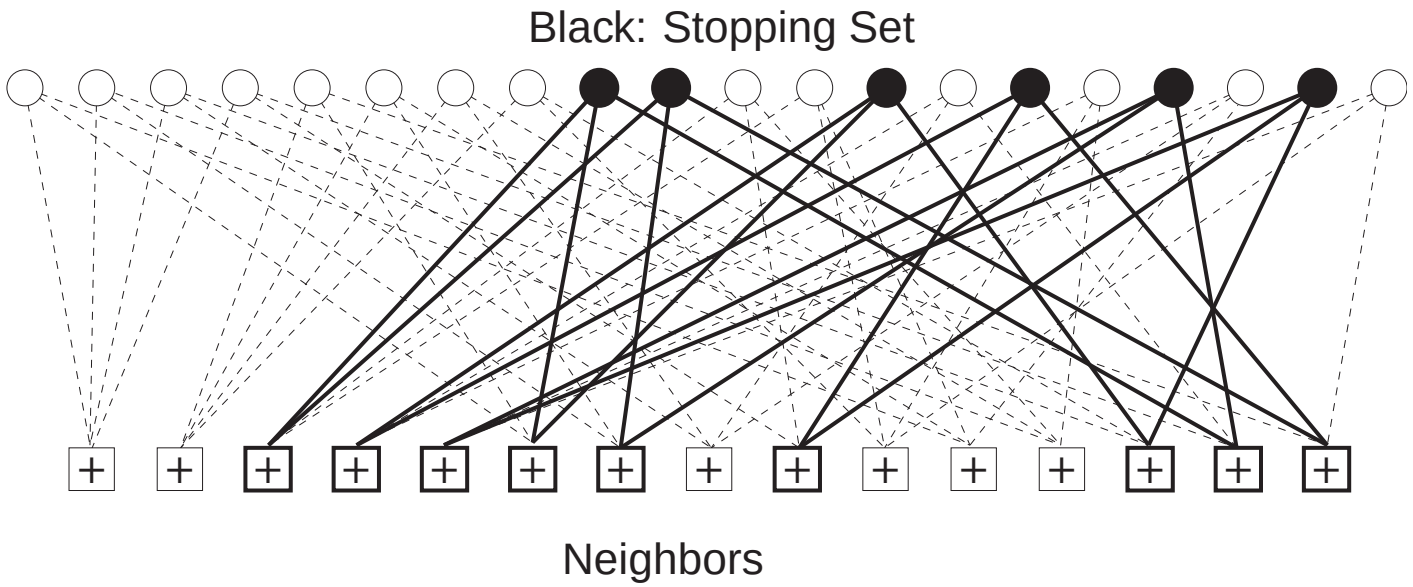


Failure of LDPC on the BEC

Large LDPCs are extremely effective on the BEC channel. The erasure patterns that a code can not recover are all well defined they are related to **Stopping Sets**

A stopping set $\mathcal{S}$ is a set of variable nodes, all of whose neighboring check nodes are connected to this set at least twice.

This figure shows a stopping set in our original LDPC code:

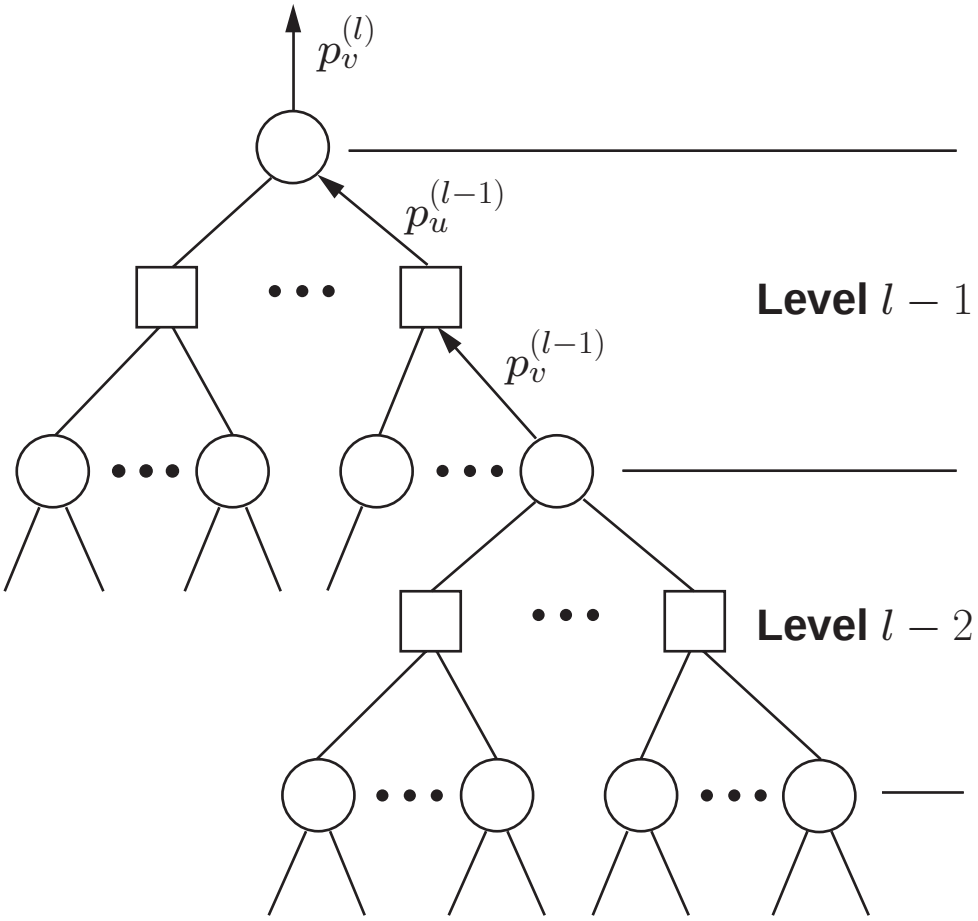


It is easy to see that if the bits in a stopping set are erased, the decoding algorithm stops, since the check node operations can not proceed.

Erasure decoding will terminate at the largest stopping set contained in the erasure set.

Probability Propagation Analysis

Assume that the code is infinitely large and has therefore no cycles:



Iterations start with an erasure probability of $p_0 = \varepsilon$ for each variable node. From this, the erasure of the outgoing message at a variable node is given by:

$$p_v^{(l)} = p_0 \left[p_u^{(l-1)} \right]^{d_v - 1}$$

The probability of a sending an erasure message from a check node is given by:

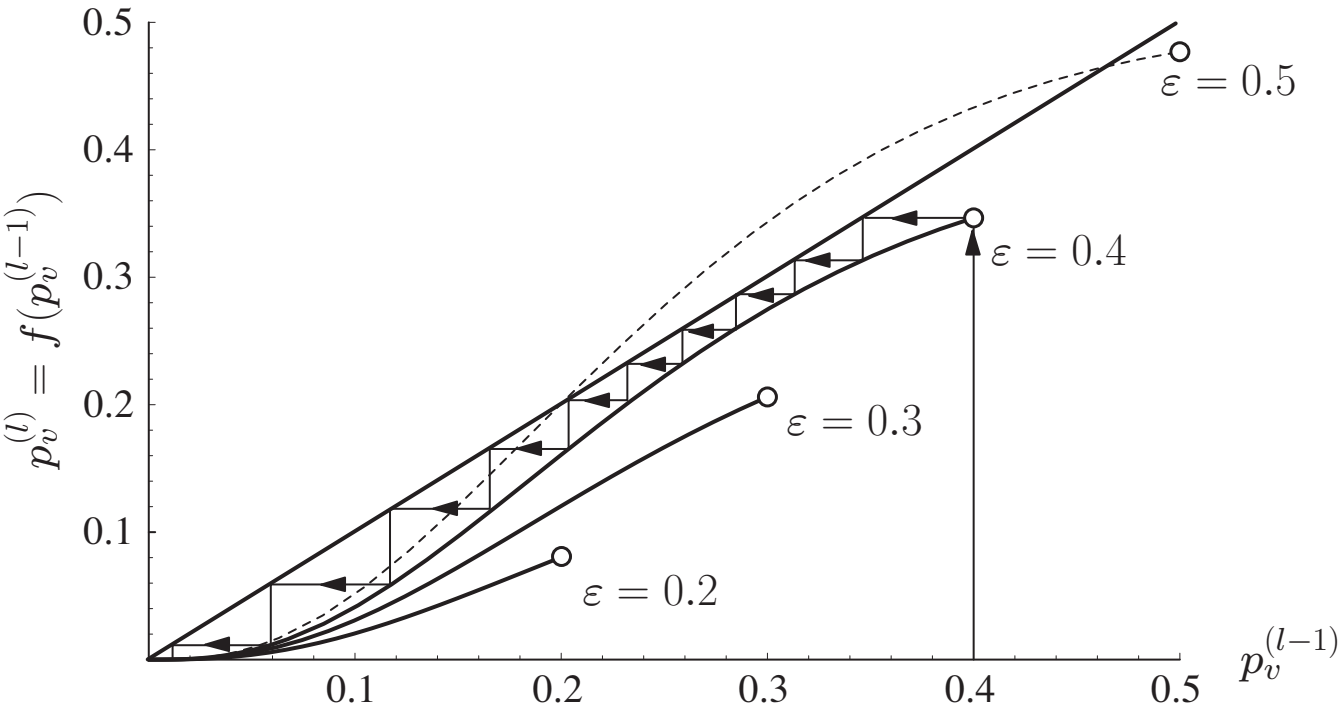
$$p_u^{(l-1)} = 1 - \left[1 - p_v^{(l-1)} \right]^{d_c - 1}$$

Probability Propagation on the BEC

From these equations we obtain the iteration formula:

$$p_v^{(l)} = p_0 \left(1 - \left[1 - p_v^{(l-1)}\right]^{d_c - 1}\right)^{d_v - 1}$$

Example: Probability propagation on a (6,3) $R = 1/2$ regular code:



For irregular LDPC codes the probability update formulas have to be modified to

$$p_u^{(l-1)} = 1 - \sum_{i=1}^{d_c} \rho_i \left[1 - p_v^{(l-1)}\right]^{i-1} = 1 - \rho\left(1 - p_v^{(l-1)}\right)$$

$$p_v^{(l)} = p_0 \sum_{j=1}^{d_v} \lambda_j \left[p_u^{(l-1)}\right]^{j-1} = p_0 \lambda\left(p_u^{(l-1)}\right)$$

Threshold of LDPCs

From these observation, a threshold parameter can be defined as

$$\begin{aligned} \varepsilon^* &= \sup\left\{\varepsilon \,:\, f\left(\varepsilon,\, x\right) < x,\, \forall\, x \leq \varepsilon\right\} \\ \text{where } f(\varepsilon,\, x) &= \varepsilon\lambda\left(1-\rho\left(1-x\right)\right) \end{aligned}$$

that is, the transfer function $f(\varepsilon, x)$ must lie entirely below the 45^0 symmetry line. Error-free decoding is possible if and only if

$$x = \varepsilon\lambda\left[1-\rho\left(1-x\right)\right]$$

has no positive solutions for $x \leq \varepsilon$.

The threshold can be rewritten as:

$$\varepsilon^* = \min\left\{\varepsilon(x) \,:\, \varepsilon(x) \geq x\right\}$$
$$\varepsilon(x) = \frac{x}{\lambda\left[1-\rho\left(1-x\right)\right]}$$

For **regular LDPC codes** we can specify this further to:

$$\varepsilon^* = \frac{1-s}{\left(1-s^{d_c-1}\right)^{d_v-1}}$$

where s is the positive real root of

$$\left[(d_v-1)(d_c-1)-1\right] y^{d_c-2} - \sum_{i=0}^{d_c-3} y^i = 0$$

The threshold for (3,6) codes is $\varepsilon^* = 0.4294$ and capacity is at $\varepsilon \geq 0.5$.

Density Evolution for the AWGN Channel

The situation in the additive white Gaussian noise channel is somewhat more complicated. The received signal LLR is given

$$f_Y(y) = \sqrt{\frac{N_0}{16\pi}} e^{-\frac{N_0}{16}\left(y-\frac{4}{N_0}\right)^2}$$

The PDF of the channel LLR is Gaussian distributed with $m_Y = 4/N_0$ and variance $2m_Y$. Such a Gaussian PDF is called **consistent** – a single parameter suffices to characterize the entire PDF.

Variable Node Processing

At the variable nodes signals are added and sent back to the check nodes. Adding Gaussian signals produces a Gaussian signal. The mean of the signal PDF that is sent to the check node is:

$$m_v^{(l)} = m_v^{(0)} + (d_v - 1) m_u^{(l-1)}$$

Check Node Processing

The situation here is a little more difficult: First, assuming the independence of the tree, we obtain for the outgoing check node message:

$$E\left[\tanh\left(\frac{U}{2}\right)\right] = E\left[\tanh\left(\frac{V_i}{2}\right)\right]^{d_c-1}.$$

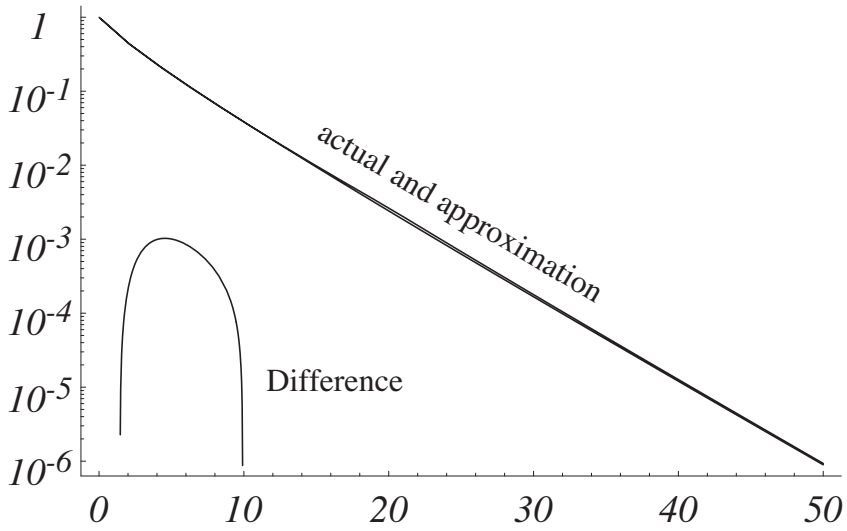
We need the following definition

$$\begin{aligned}\phi(m_u) &= 1 - E\left[\tanh\left(\frac{U}{2}\right)\right] \\ &= 1 - \frac{1}{4\pi m_u}\int_{\mathcal{R}} \tanh\left(\frac{u}{2}\right)\exp\left[-\frac{1}{4m_u}(u-m_u)^2\right] du\end{aligned}$$

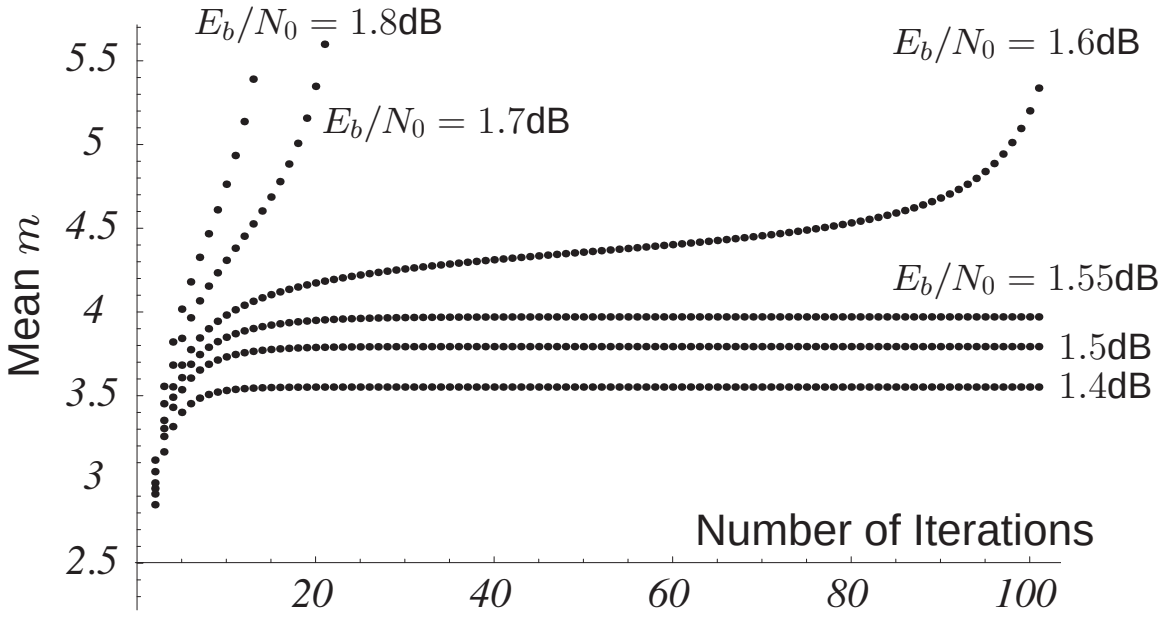
Check Node Transfer Functions

Function $\phi(m)$ is a non-elementary integral. It does have close approximations which speed up the computations substantially.

$$\phi(m) \approx \begin{cases} \exp\left(-0.4527 m^{0.86} + 0.0218\right); & \text{for } m < 19.89 \\ \sqrt{\frac{\pi}{m}} \exp\left(-\frac{m}{4}\right)\left(1 - \frac{1}{7m}\right); & \text{for } m \geq 19.89 \end{cases}$$



An infinite-size code converges if m_u diverges to ∞ as the number of iterations increases (Example for the (4,8) regular LDPC)



Numerical Issues

The divergence to ∞ is somewhat cumbersome. We observe that $\phi(m) \to 0$ as $m \to \infty$. Let's define $r = \phi(m_u^{(l-1)})$, and

$$h(s,\, r) = \phi\left[s + (d_v - 1)\,\phi^{-1}\left(1 - (1-r)^{d_c - 1}\right)\right]$$

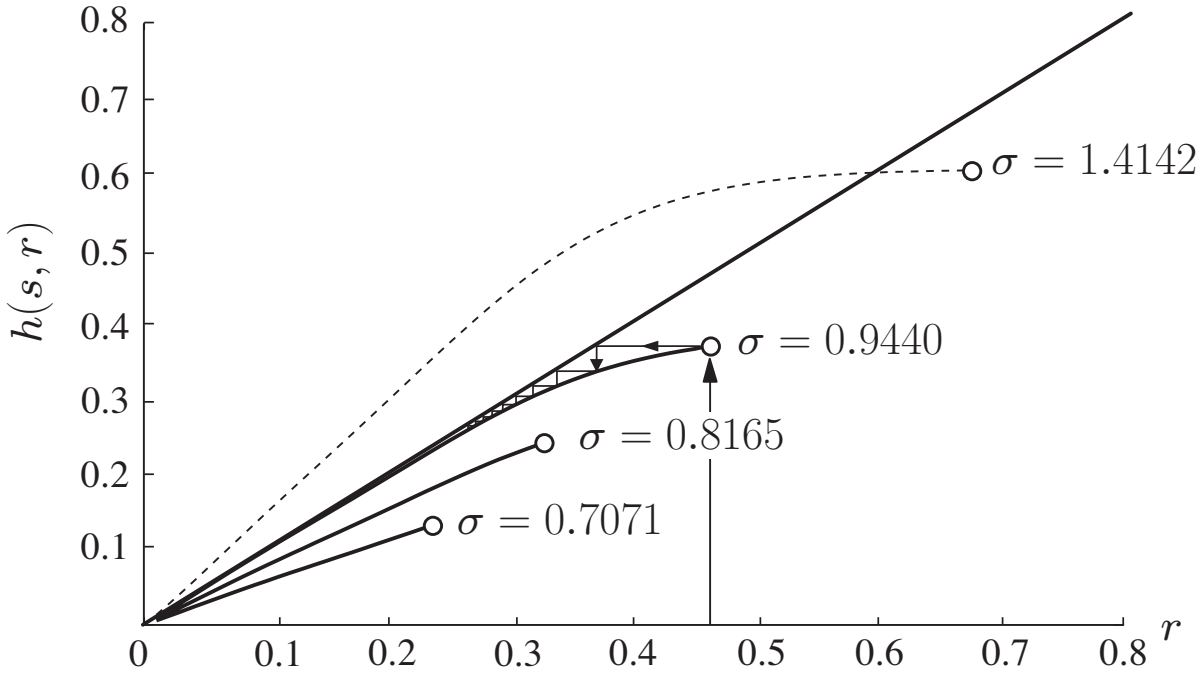
where we note that $h(s,\, \phi(m_u^{(l-1)})) = \phi(m_u^{(l)})$, $s = 4/N_0$.

We now have a convergence to zero situation, which is analogous to the probability convergence for the BEC just discussed.

The threshold is therefore defined as

$$s^* = \inf\left\{s \in \mathcal{R}^+ \,:\, h(s,\, r) - r < 0,\, \forall\, r \in (0,\, \phi(s))\right\}$$

and the threshold noise variance is: $\sigma^* = \sqrt{\frac{2}{s^*}}$



LDPC Irregular Code Analysis

Density analysis for irregular code is essentially an extension of the above analysis with a few noteworthy differences:

Variable Nodes:
Due to the irregularity, the messages **leaving** the variable are a Gaussian mixture with means for a node with degree i given by

$$m_{v,i}^{(l-1)} = (i-1)m_u^{(l-1)} + m_v^{(0)}$$

Check Nodes:
The signals entering the check nodes are Gaussian mixtures, and the check node output signal is obeys for a node of degree j

$$\begin{aligned} E\left[\tanh\left(\frac{U}{2}\right)\right] &= \prod_{i=1}^{j-1} E\left[\tanh\left(\frac{V_i}{2}\right)\right] \\ \phi\left(m_{u,j}^{(l)}\right) &= 1 - \left[1 - \sum_{i=1}^{d_v} \lambda_i \phi\left((i-1)m_u^{(l-1)} + m_v^{(0)}\right)\right]^{j-1} \end{aligned}$$

The **average** check node output signal is then simply

$$m_u^{(l)} = \sum_{j=1}^{d_c} \rho_j \phi^{-1}\left(1 - \left[1 - \sum_{i=1}^{d_v} \lambda_i \phi\left((i-1)m_u^{(l-1)} + m_v^{(0)}\right)\right]^{j-1}\right)$$

This is recursive formula for m_u.

Note The check node output signal may not be exactly Gaussian, but these signals are mixed by the additive variable node which produces a Gaussian with high accuracy, especially if d_v is large.

Success of Irregular LDPCs

d_v	4	8	9	10	11	12	15	20	30	50
λ_2	.3835	.3001	.2768	.2511	.2388	.2443	.2380	.2199	.1961	.1712
λ_3	.0424	.2840	.2834	.3094	.2952	.2591	.2100	.2333	.2404	.2105
λ_4	.5741			.0010	.0326	.0105	.0349	.0206		.0027
λ_5						.0551	.1202			
λ_6								.0854	.0023	
λ_7							0159	.0654	.0552	.0001
λ_8		.4159				.0146		.0477	.1660	.1527
λ_9			.4397					.0191	.0409	.0923
λ_{10}				.4385		.0128			.0106	.0280
λ_{11}					.4334					
λ_{12}						.4037				
λ_{14}							.0048			
λ_{15}							.3763			.0121
λ_{19}								.0806		
λ_{20}								.2280		
λ_{28}									.0022	
λ_{30}									.2864	.0721
λ_{50}										.2583
ρ_5	.2412									
ρ_6	.7588	.2292	.0157							
ρ_7		.7708	.8524	.6368	.4301	.2548				
ρ_8			.1319	.3632	.5699	.7344	.9801	.6458	.0075	
ρ_9						.0109	.0199	.3475	.9910	.3362
ρ_{10}								.0040	.0015	.0888
ρ_{11}										.5750
σ^*	.9114	.9497	.9540	.9558	.9572	.9580	.9622	.9649	.9690	.9718
$\frac{Eb}{N_0}$	0.806	0.448	0.409	0.393	0.380	0.373	0.335	0.310	0.274	0.248
σ^*_{GA}	.9072	.9379	.9431	.9426	.9427	.9447	.9440	.9460	.9481	.9523
$\frac{Eb^*}{N_0}$	0.856	0.557	0.501	0.513	0.513	0.494	0.501	0.482	0.462	0.423

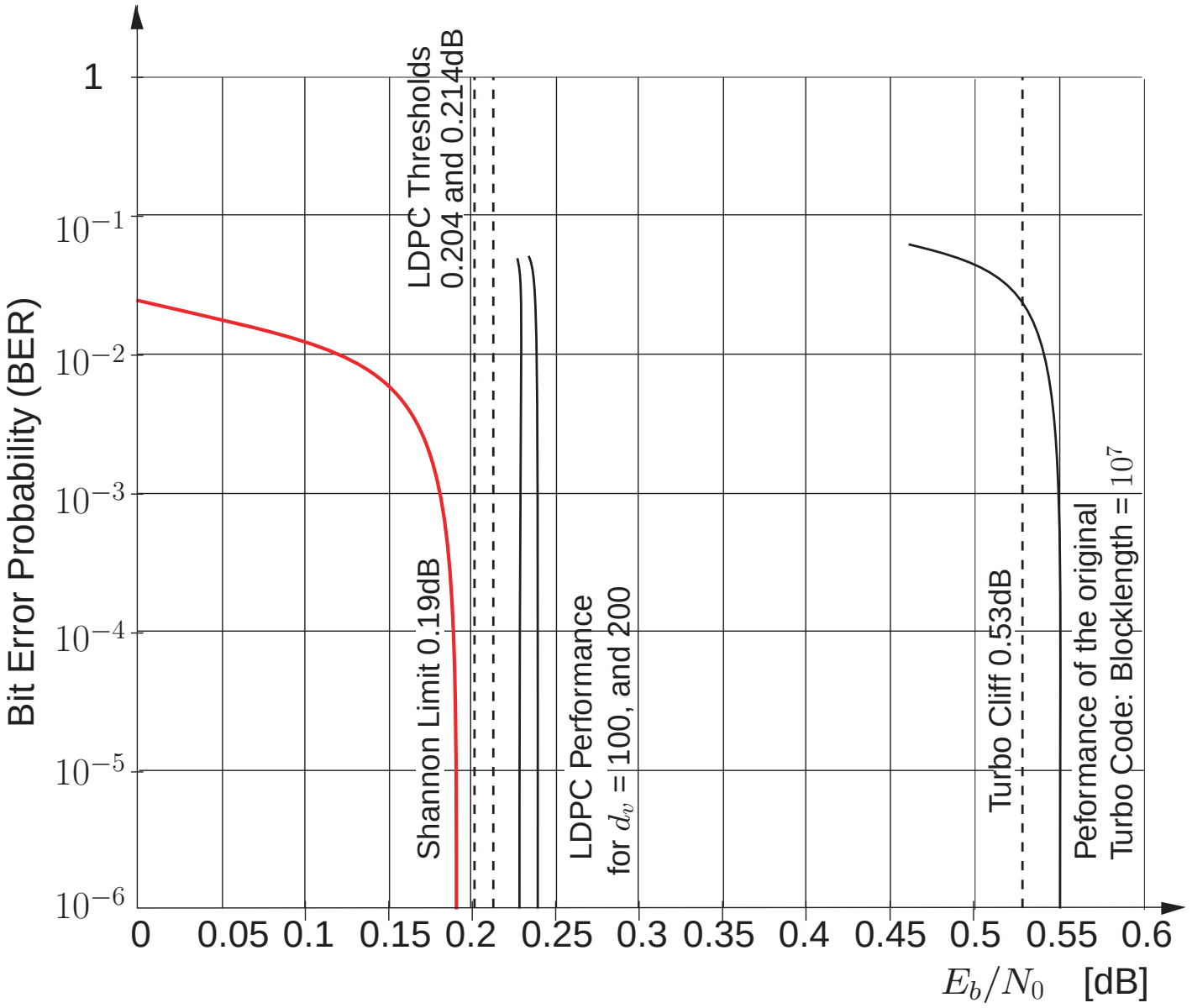
Capacity lies at $\sigma^2 = 0.9787$ corresponding to $E_b/N_0 = 0.188$dB.

Check Node Concentration means that

$$\rho(x) = \rho_k x^{k-1} + (1 - \rho_k)\, x^k.$$

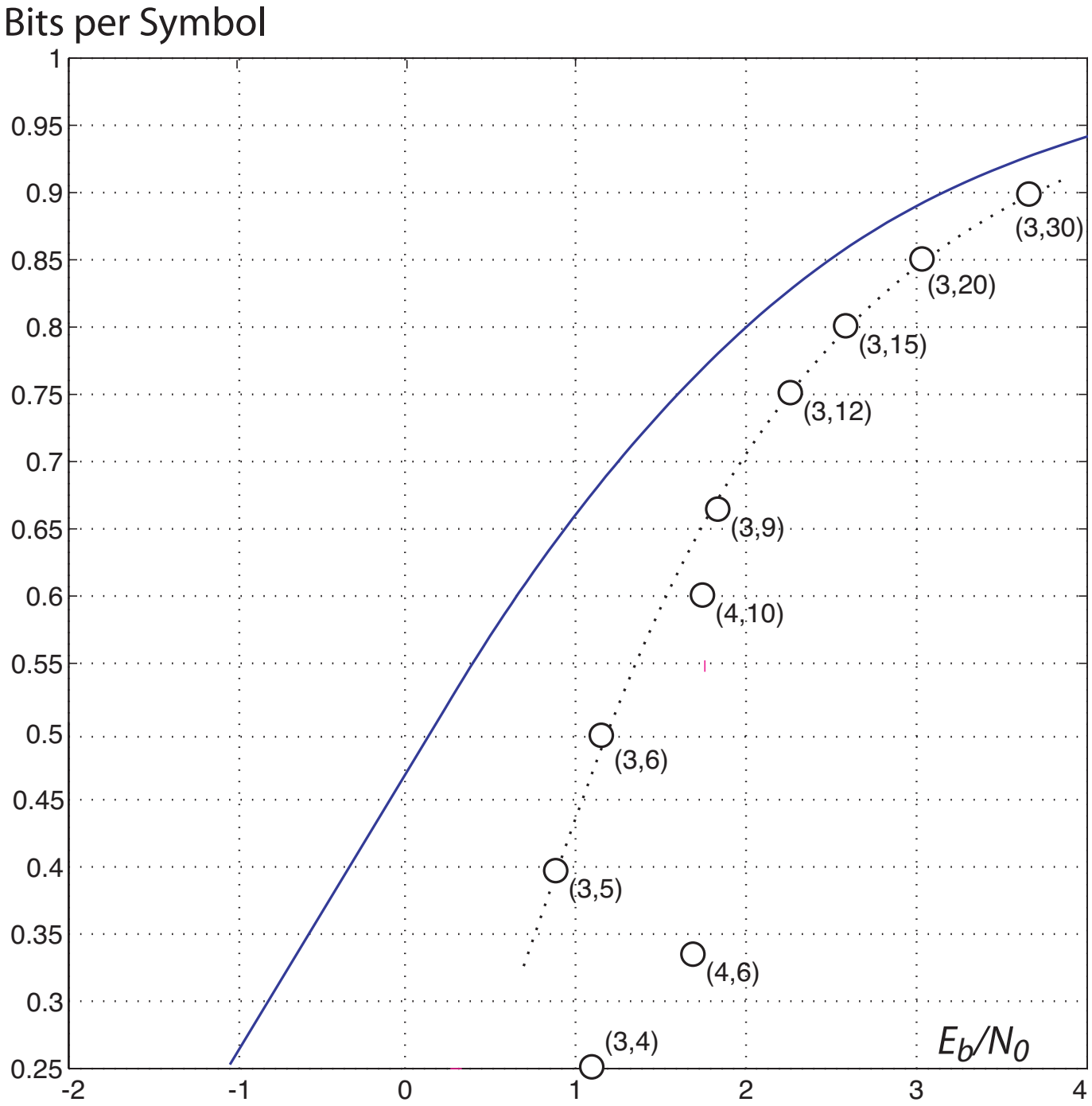
Very Large LDPC Codes

Construction and simulations of very large LDPC codes reveal close to Shannon limit performance:



[Chun01]	S.Y. Chung, G.D. Forney, T.J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045dB of the Shannon limit," *IEEE Comm. Lett.,* vol. 5, no. 2, pp 58–60, February 2001.

Limited Performance of Regular Codes

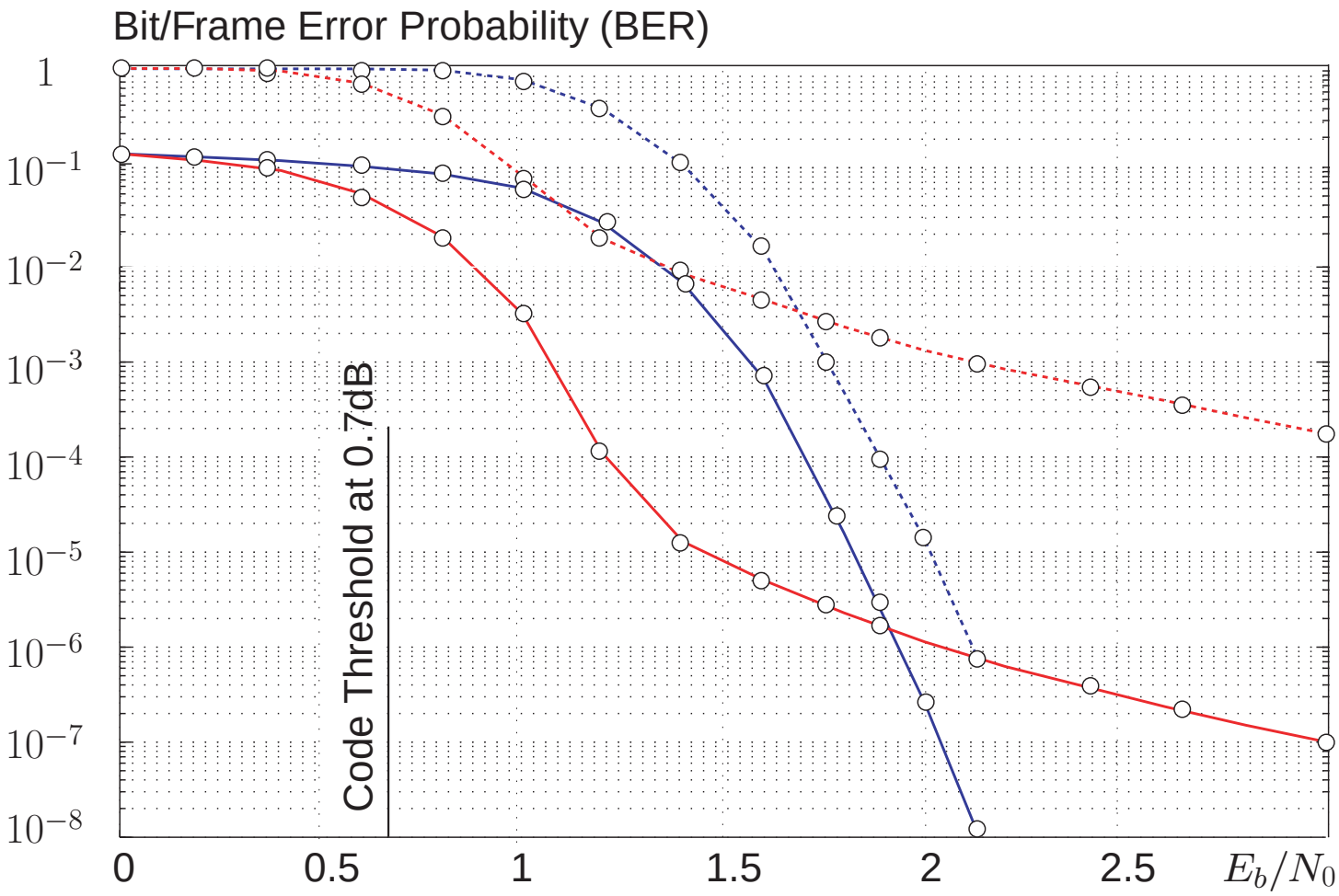


- Threshold convergence of regular LDPC codes is close to the BPSK capacity limit for high rates.
- Regular codes perform poorly at lower rates $\Rightarrow$ Irregular codes

[Schl034] C. Schlegel and L. Perez, *Trellis and Turbo Coding*, IEEE/Wiley, 2004, also: `www.turbocoding.net`: LDPC Chapter.

Error Floor Phenomenon

Performance results for regular and irregular cycle optimized LDPC codes of rate $R = 1/2$ for a block length of 4000 bits



Like Turbo Codes, (randomly) constructed LDPC codes suffer from an error floor which is difficult to determine analytically. We observe:

- **Irregular Codes:** have a higher error floor
- **Regular Codes:** have typically a lower error floor, but less performance in the waterfall region
- **LDPC Codes:** rarely fail (decode erroneously) to a codeword

Counter Measures

There have been a number of strategies to lower the error floor:

- **Increasing Girth:** This increases the length of the shortest cycles which have been implicated in correlating the messages in the iterative decoder.

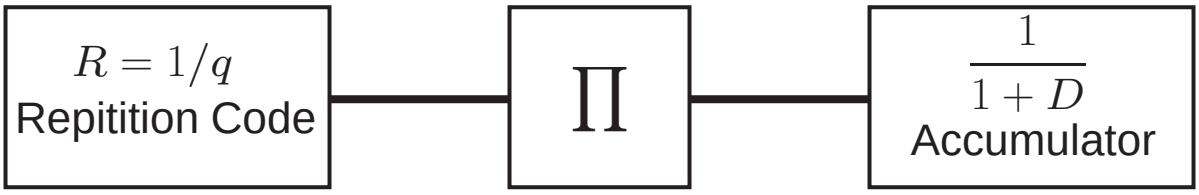
- **Special Construction:** LPDC codes constructed on expander graphs have provably large girths, but their rates and performance in the threshold region tend to be problematic

- **Triangular and Repeat Accumulate Structures:** Relegating variable nodes with low degrees to be parity checks has strong impact. Low degree variable nodes tend to have higher error rates.

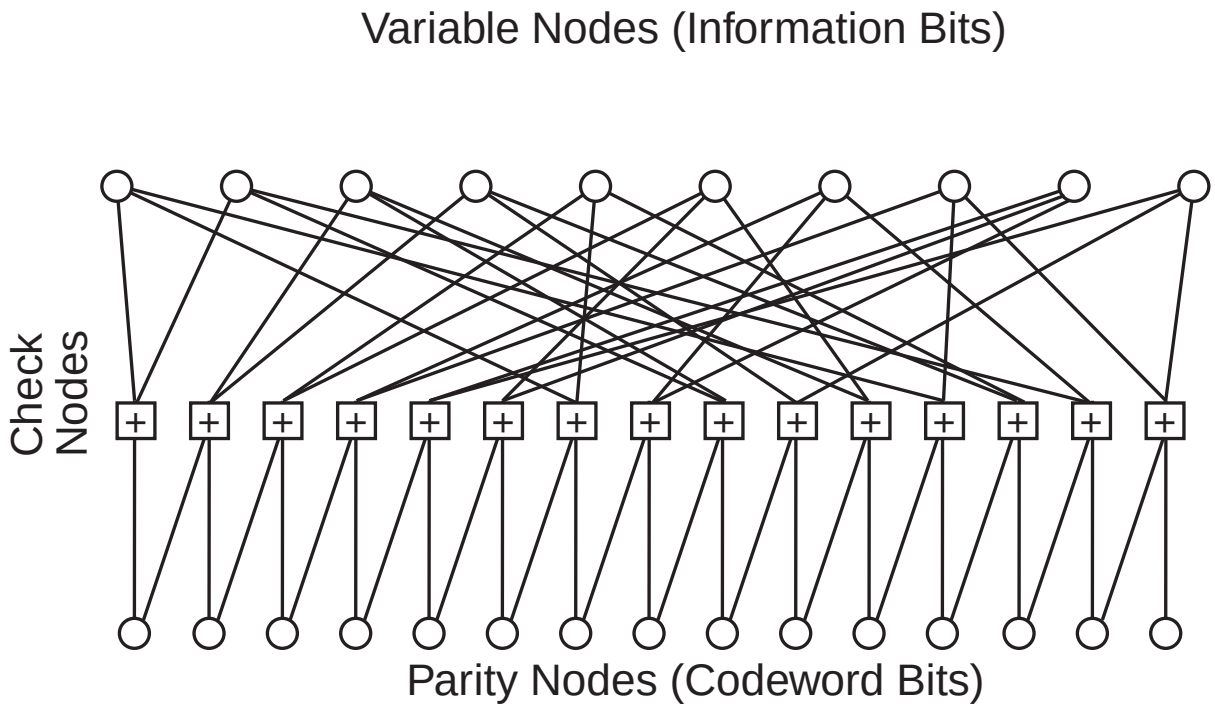
- **Increasing the Extrinsic Message Degree of Short Cycles:** This method is a combination of girth and a method to insure influx of sufficient extrinsic information from other parts of the code graph. The resulting construction – Approximate cycle EMD, or ACE, produces low error floor LDPC codes.

Repeat-Accumulate Codes

RA codes are really serially concatenated turbo codes where the outer code is a very similar repetition code:

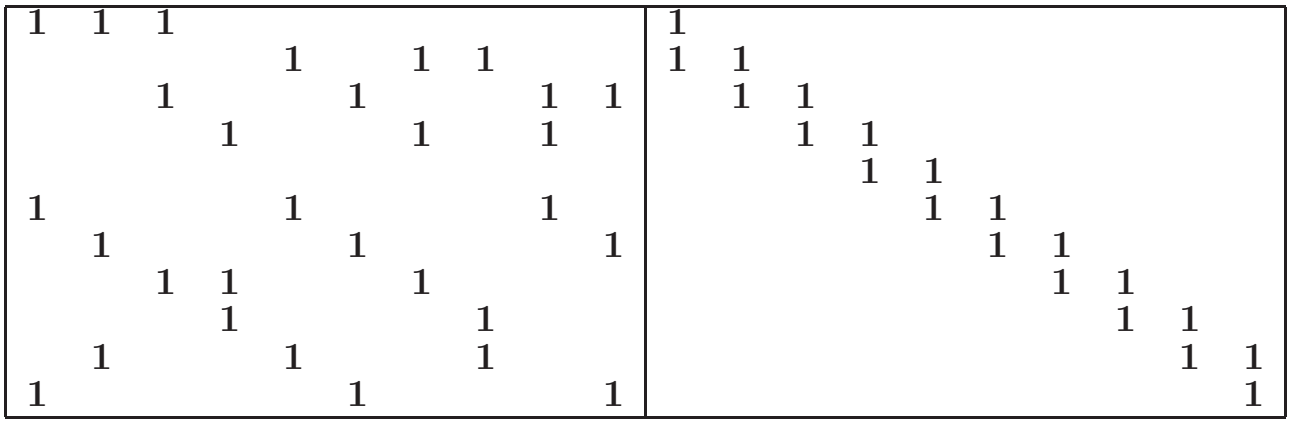


However, if we draw the code graph of a repeat accumulate code, we see that it can just as well be interpreted as a low-density parity check code where the parity checks are degree-2 nodes which can be **recursively** encoded (from right to left).



Irregular Repeat-Accumulate Codes

The parity-check matrix of a repeat accumulate code reflects the accumulator structure in the parity portion of the matrix:



Irregular RA codes can be optimized for degree distributions also:

a	2	3	4
λ_2	.139025	.078194	.054485
λ_3	.222155	.128085	.104315
λ_5		.160813	
λ_6	.638820	.036178	.126755
λ_{10}			.229816
λ_{11}			.016484
λ_{12}		.108828	
λ_{13}		.487902	
λ_{27}			.450302
λ_{28}			.017842
Rate	0.333364	0.333223	0.333218
σ^*	1.1981	1.2607	1.2780
σ_{GA}	1.1840	1.2415	1.2615
Capacity (dB)	-0.4953	-0.4958	-0.4958

Their rate is given as $R = a/(a + \sum_i i\lambda_i)$.

Extended Irregular Repeat Accumulate Codes

Yang et. al. [YanRya04] have constructed such codes using the optimal degree distributions for a number of rates. They have found that by increasing the column weight in the information portion of the parity-check matrix they could improve the error floor.

Example: (4161,3430) eIRA codes constructed

Code 1: $\lambda(x) = 0.00007 + 0.1014x + 0.5895x^2 + 0.1829x^6 + 0.1262x^7$

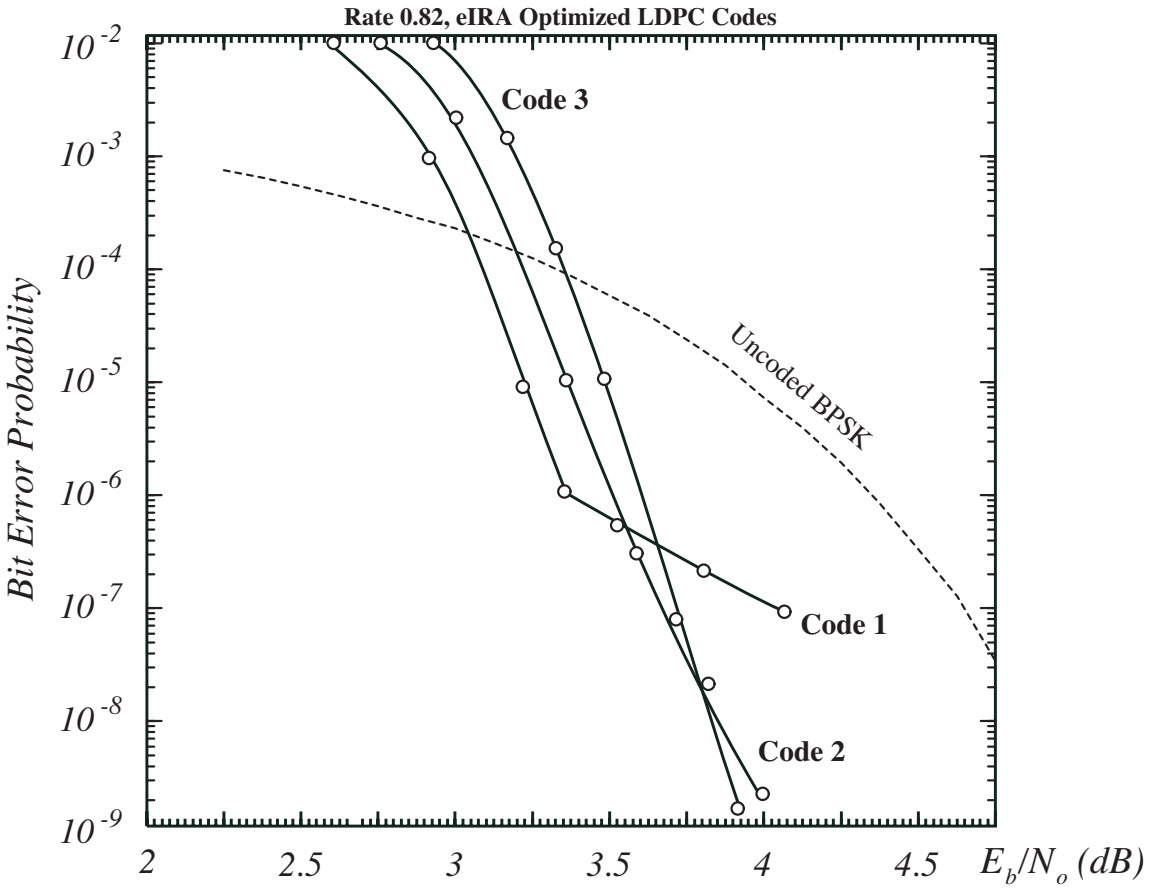
$\rho(x) = 0.3037x^{18} + 0.6963x^{19}$

Code 2: $\lambda(x) = 0.0000659 + 0.0962x + 0.9037x^3$

$\rho(x) = 0.2240x^{19} + 0.7760x^{20}$

Code 3: $\lambda(x) = 0.0000537 + 0.0784x + 0.9215x^4$

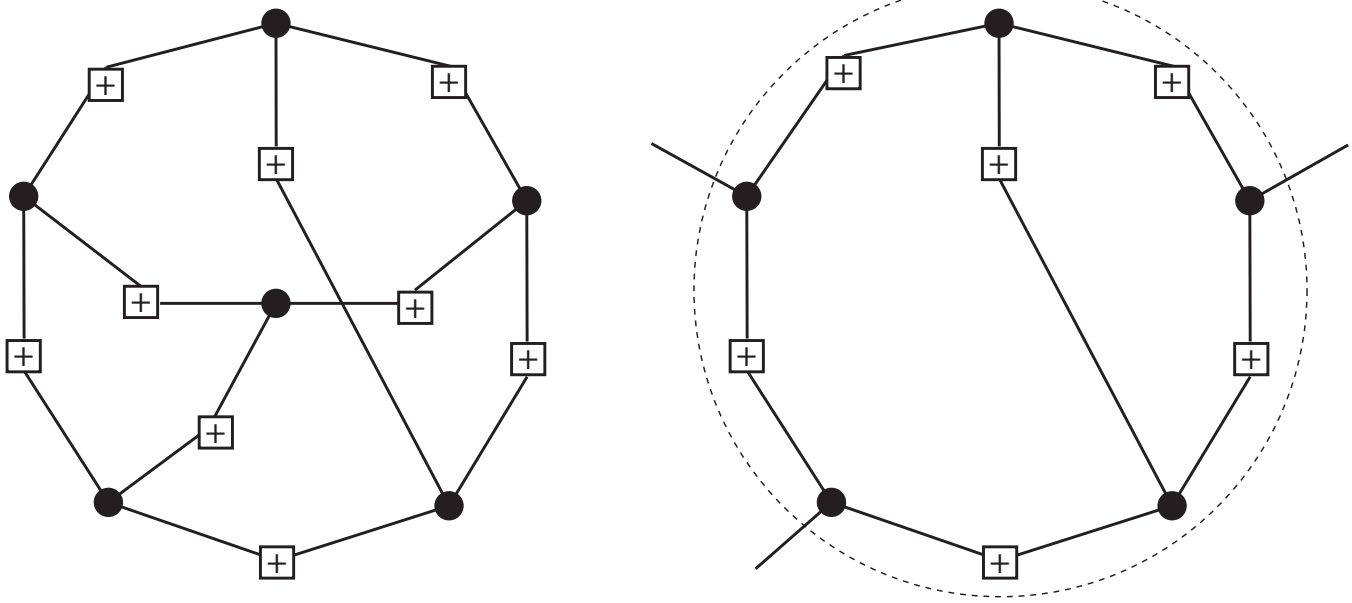
$\rho(x) = 0.5306x^{24} + 0.4694x^{15}$



[YanRya04] M. Yang, W.E. Ryan, and Y. Li, "Desin of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 564–571, April 2004.

ACE Construction Algorithm

Extrinsic Message Degree (EMD) of a Set is defined as the number of connections from variable nodes of the set to the "outside":



Stopping Set, EMD = 0 Smaller Set, EMD = 3

A Stopping Set has an EMD of zero. No outside edges join the variable nodes.

Approximate Cycle EMD (ACE) is a "practical measure", where we simply ignore intraset constraints, i.e., the set above has an ACE of 5.

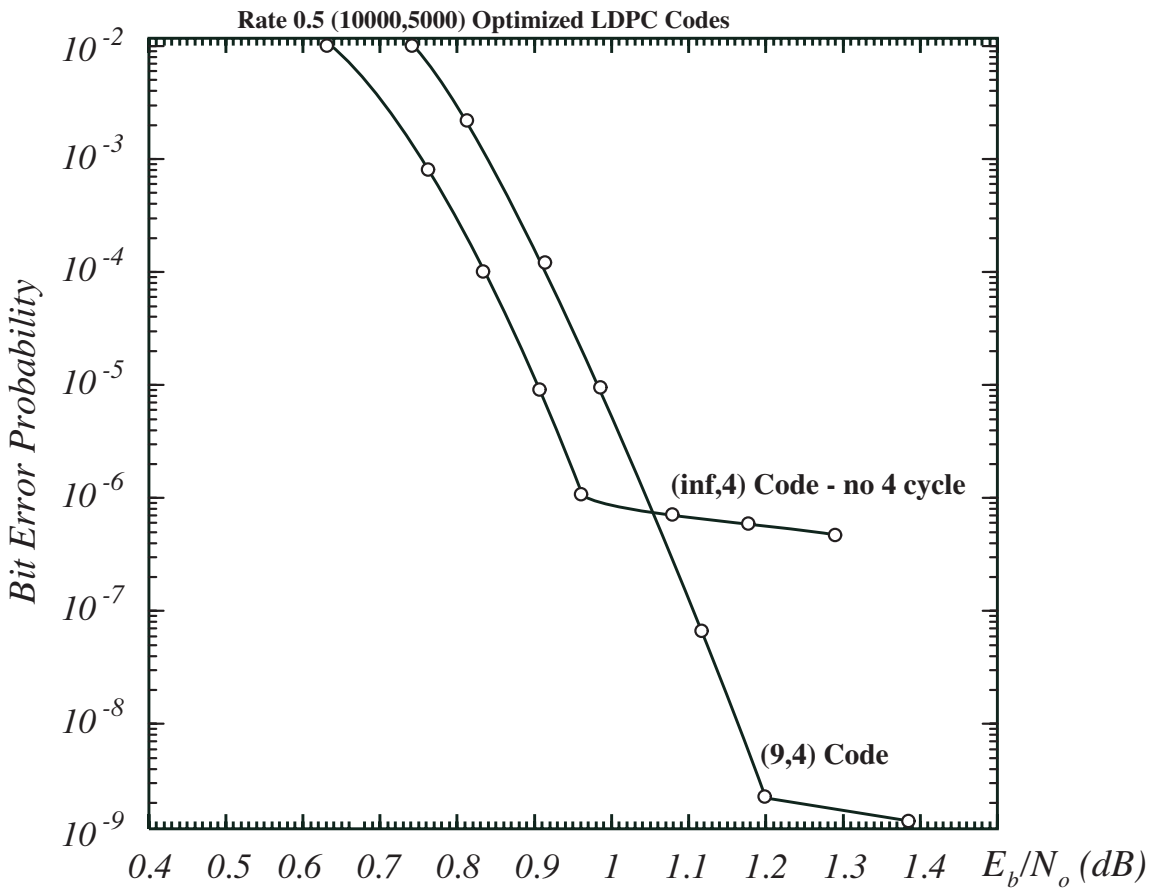
In general the ACE of a circle of length $2d$ equals

$$\mathrm{ACE} = \sum_i (d_i - 2)$$

ACE Construction of LDPCs

An LDPC has $(d_{\mathrm{ACE}}, n_{\mathrm{ACE}})$ if all cycles of length $l \leq 2d_{\mathrm{ACE}}$ have ACE $\geq n_{\mathrm{ACE}}$.

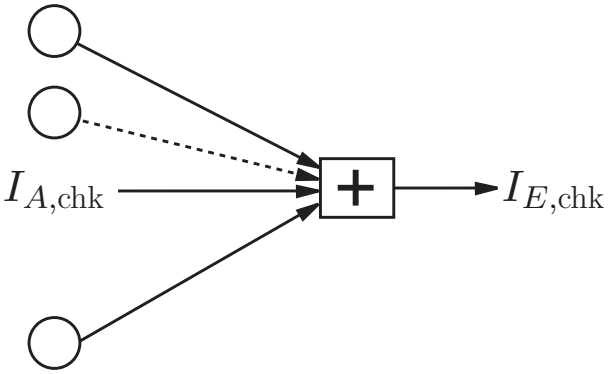
Tian et. al. [Tia03] construct such codes by randomly generating codes until a code meets the ACE criterion. Good codes can be constructed this way:



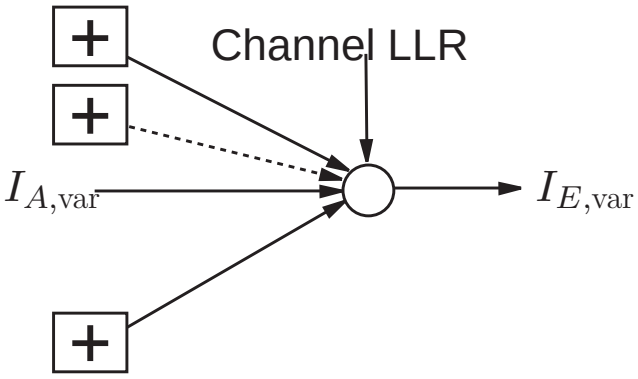
[Tia03]	T. Tian, C. Jones, J.D. Villasenor, R.D. Wesel, "Selective avoidance of cycles in irregular LDPC code construction," *IEEE Trans. Commun.*, submitted.

LDPC Code Design via EXIT Charts

There have also been efforts to design LDPC codes via EXIT analysis. EXIT is similar to density evolution:

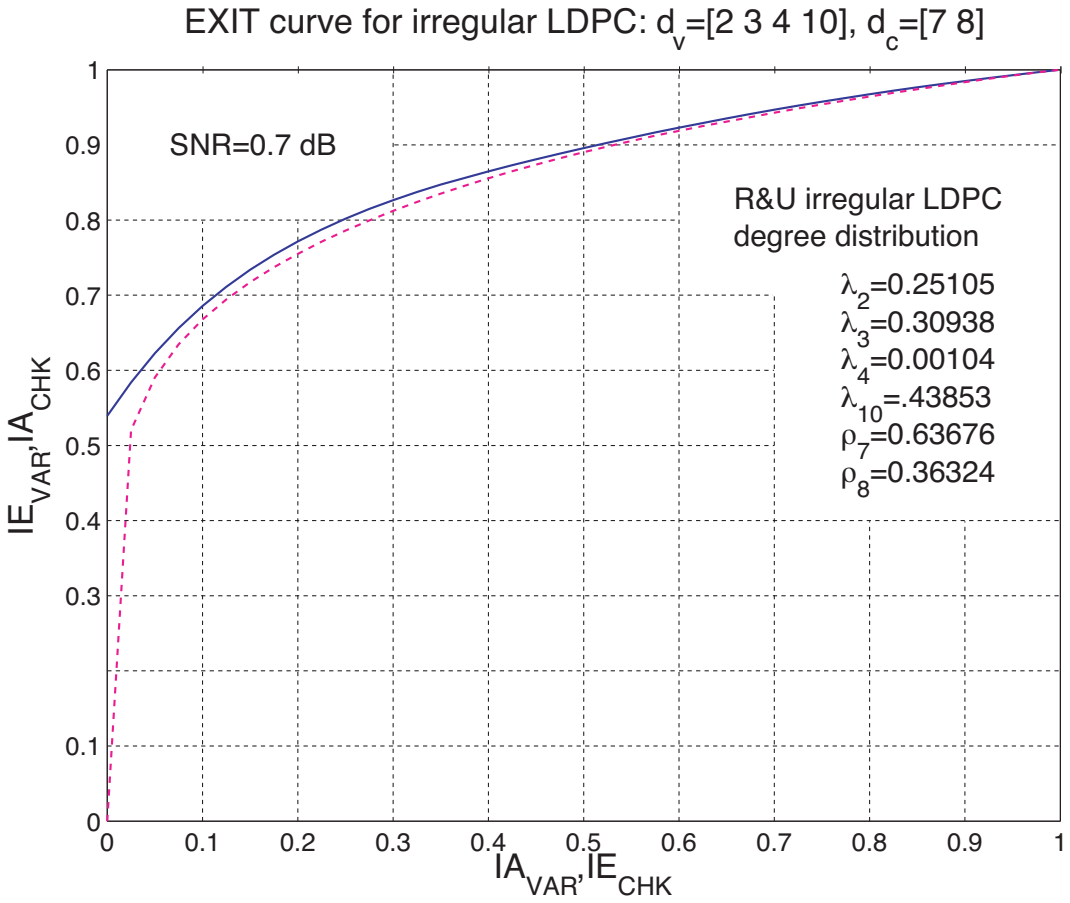


Parity Check Code



"Repetition Code"

The following code parameters were designed by Howard et. al. and show a high-performing irregular LDPC code:

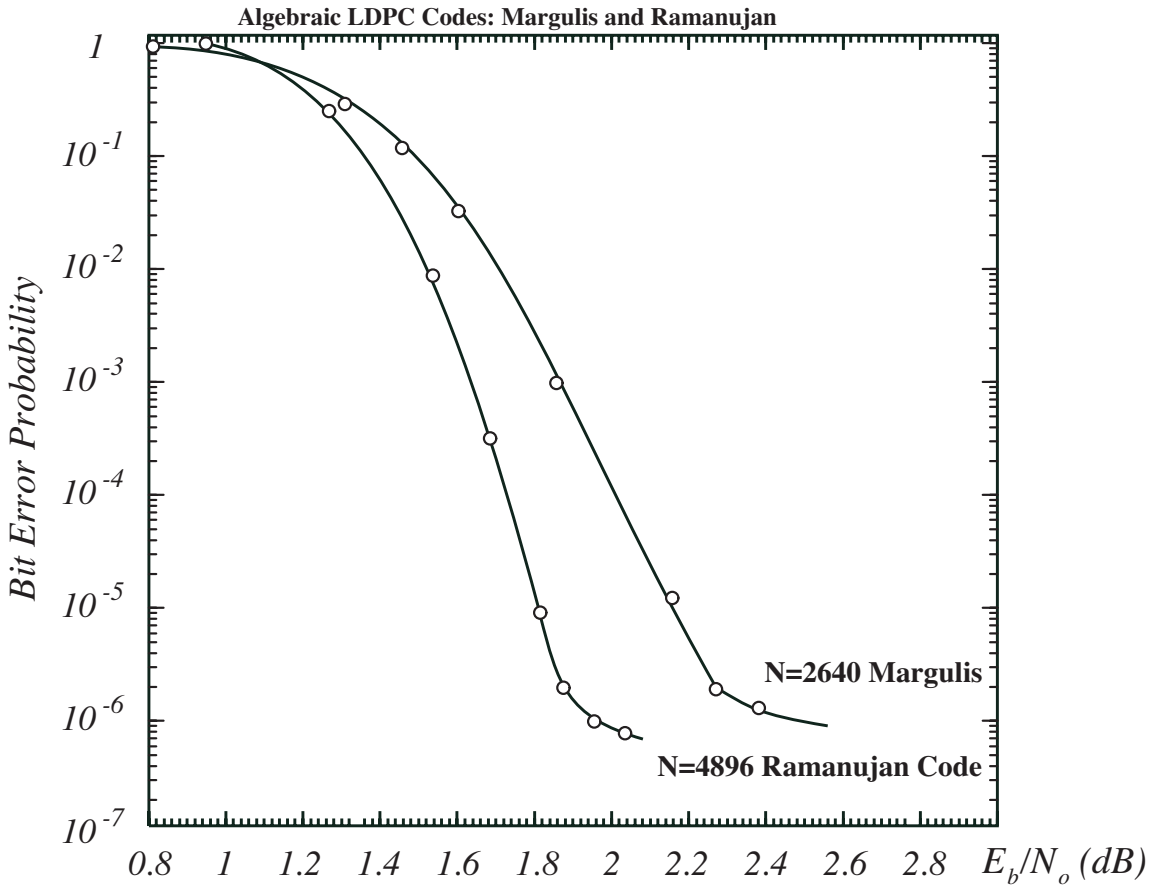


Specialized Designs

Construction of Margulis [Mar82] produces codes of length $N = 2(p^2-1)p$ codes, for each prime p with a girth which grows as $\log p$.

Ramanujan Graphs have small second eigenvalues of their adjacency matrix which guarantees large girths.

For $p = 11$, the resulting has girth 8 and $N = 2640$.



[Mar82]	G.A. Margulis, "Explicit construction of graphs without short cycles and low-density parity check codes," *Combinatorica*, vol. 2, no. 1, pp. 71-78, 1982.

Problems with Algebraic Constructions

The Margulis Code suffers from decoding failure due to near-codewords: A Hamming weight w sequence which causes a weight v parity check violation is called a (w, v) near-codeword. The offending near code-words are (12,4) adn (14,4) near-code words.

The Ramanujan Code has weight-24 actual codewords, which are low-weight enough to cause the error floor.

In General: Algebraic Constructions are problematic also:

- A large girth does not guarantee a low error floor under iterative decoding
- Codes may have low weight codewords even though they have large girth
- Constructions usually generate only codes with few and very specific parameters such as length, rates, etc.

[1] [MaPo03] D.J.C. MacKay and M.S. Postol, "Weaknesses of Margulis and Ramanujan-Margulis low-density parity-check codes," *Electronic Notes Theor. Comp. Sci.,* vol. 74, 3003.

The Encoding Problem

- In general, encoding of **linear codes** is accomplished by finding the generator matrix $\mathbf{G}$:

$$\mathbf{v} = \mathbf{u}\mathbf{G}$$

- To find $\mathbf{G}$, the parity check matrix is first put in systematic form (using Gaussian elimination techniques) and then

$$\mathbf{H} \rightarrow [\mathrm{tr}\,\mathbf{P}|\mathbf{I}_{N-K}] \rightarrow \mathbf{G} = [\mathbf{I}_K|\mathbf{P}]\,.$$

- **Example:** Consider a $(10, 3, 5)$ LDPC code with

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \Rightarrow \left[\begin{array}{c|cccc} & 1 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 1 \\ \mathbf{I}_6 & 0 & 0 & 1 & 0 \\ & 1 & 1 & 1 & 1 \\ & 1 & 1 & 1 & 1 \\ & 0 & 1 & 0 & 0 \end{array}\right]$$

Thus,

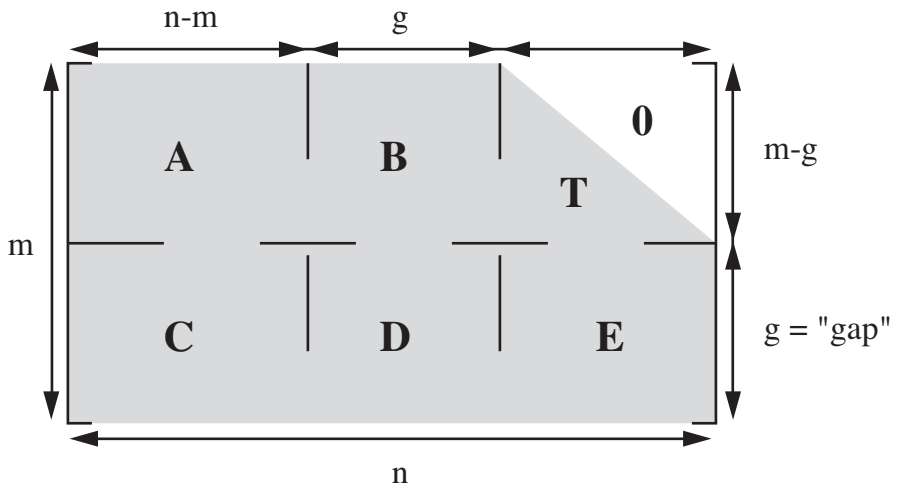
$$\mathbf{G} = \left[\begin{array}{c|cccccc} \mathbf{I}_4 & 1 & 0 & 0 & 1 & 1 & 0 \\ & 0 & 0 & 0 & 1 & 1 & 1 \\ & 0 & 0 & 1 & 1 & 1 & 0 \\ & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right]$$

In general, $\mathbf{G}$ is no longer sparse, and due to the matrix multiplication, the encoding complexity of LDPC codes is $O(N^2)$.

LInear-Time Encoding

Ideally, we would wish to have a **triangular** parity-check matrix, in which case encoding could be performed via simple successive back-substitution.

An **approximate triangularization** has been used by Richardson and Urbanke [RiUr01] of the form



Split $\mathbf{H}$into $[\mathbf{H}_u \mid \mathbf{H}_{*p}]$, giving the equation

$$\mathbf{H}_p \mathbf{x}_p^T = \mathbf{H}_u \mathbf{x}_u^T \Rightarrow \mathbf{x}_p^T = \mathbf{H}_p^{-1} \mathbf{H}_u \mathbf{x}_u^T.$$

The parity-check rule then gives the following encoding equations

$$\begin{aligned} \mathbf{A}\mathbf{x}_u^T + \mathbf{B}\mathbf{p}_1^T + \mathbf{T}\mathbf{p}_2^T &= \mathbf{0}, \\ (\mathbf{C} - \mathbf{E}\mathbf{T}^{-1}\mathbf{A})\mathbf{x}_u^T + (\mathbf{D} - \mathbf{E}\mathbf{T}^{-1}\mathbf{B})\mathbf{p}_1^T &= \mathbf{0}. \end{aligned}$$

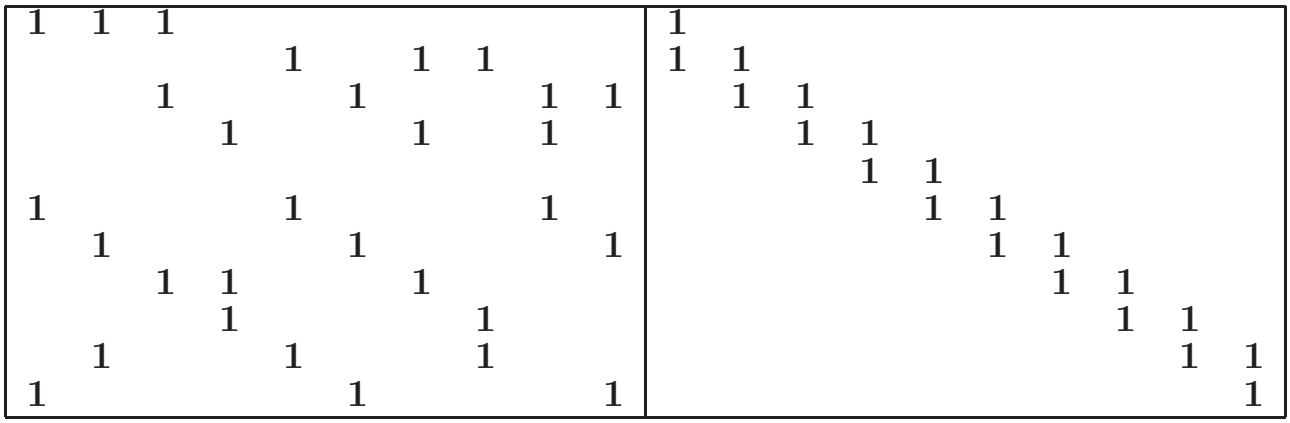
Define $\phi = \mathbf{D} - \mathbf{E}\mathbf{T}^{-1}\mathbf{B}$, and assume ϕ is non-singular, then:

$$\begin{aligned} \mathbf{p}_1^T &= \phi^{-1}\left(\mathbf{C} - \mathbf{E}\mathbf{T}^{-1}\mathbf{A}\right)\mathbf{x}_u^T, \\ \mathbf{p}_2^T &= -\mathbf{T}^{-1}\left(\mathbf{A}\mathbf{x}_u^T + \mathbf{B}\mathbf{p}_1^T\right). \end{aligned}$$

[RiUr01] T.J. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, pp. 638–656, February 2001.

Practical Linear-Time Encodable LDPC codes

Extended IRA LDPC Codes
have a lower triangular parity-check matrix and can be encoded using an accumulator:



Lower Triangular LDPC Codes
Imposed the lower triangular constraint on $\mathbf{H}$.

Iterative Encoding: Assign a set of $N - K$ nodes to be parity-check nodes which **does not contain** a stopping set, and use erasure decoding as the encoding mechanism. Proceed as follows. Declare the parity check as erasures, set all the information bits to their values, and use the erasure decoding algorithm to determine the parities.

Some Leading Commercial Products

Company	Code	Platform	Clock	Itera-tions	Bits/Cycle	Throughput
TrellisWare	Serial Turbo Code	Standard Cell ASIC	105 MHz	31 / 17	0.51 / 0.93	54 / 98 Mbit/s
L3	Parallel Turbo Code	Xilinx (50% Virtex Pro 70)	135 MHz	10 / 7	0.37 / 0.5	50 / 68 Mbit/s
iCoding	Parallel Duo-Binary Turbo Code	Xilinx (100% Virtex 2V4000)	68 MHz	16	1	68 Mbit/s
Blanksby& Howland	LDPC	ASIC (7.5mm x 7mm)	64MHz	64 / 32	8 or 16	512 / 1024 Mbit/s
UofAlberta (study)	LDPC	Xilinx (100% Virtex Pro 70)	75MHz	32	4 or 8	300 / 600 Mbit/s

High-Speed Decoders The important measure is the number of bits/clock cycle that can be attained.

[BlHo02] A.J. Blanksby and C.J. Howland, “A 690-mW 1-Gb/s 1024-bit, rate-1/2 low-density parity-check code decoder,” *IEEE J. Solid-State Cir.*, vol. 37, no. 3, pp. 404–412, March 2002.

LDPC: Summary Remarks

- LDPC codes can be constructed that achieve very excellent performance near the Shannon limit.
- Encoding of LDPC codes is not an complexity issue
- Controlling the error floor of LDPC is possible – even though not fully understood – via
  1. Assign low-degree variable nodes as the parity nodes; they may have high error rates
  2. Avoid short cycles
  3. Avoid short cycles with low extrinsic message degrees
- Error control codes can be efficiently built on ASIC or FPGA platforms

Turbo Codes and LDPC Codes effectively solve the channel coding problem for the additive-white Gaussian channel (and similar channels) with implementable encoders and decoders.

Analog Decoding

Analog Computation and APP

Digital implementations of APP decoders can be very complex and resource-intensive. Analog decoding provides an alternative with some attractive features:

- Parallel design provides speed and robustness under process variation.

- The APP algorithm preserves high precision at the system level in spite of reduced precision at the component level.

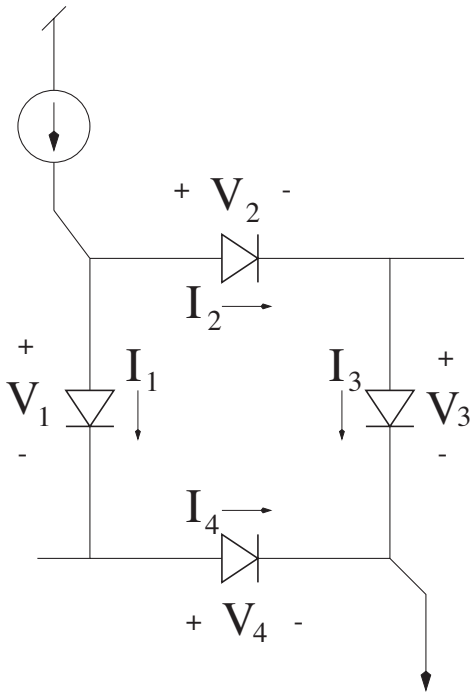
- CMOS designs in subthreshold allow fabrication using all-digital processes. Subthreshold CMOS circuits consume very little power, making analog decoding attractive for ultra-low power applications.

- Continuous-time processing replaces iteration, giving analog circuits both elegance of design and an additional degree of resource efficiency.

- Subthreshold current mode operation can substantially reduce power requirements.

Translinear Devices

- A translinear device is a voltage-controlled current-source for which the current is an exponential function of the voltage. Typical examples are diodes and bipolar transistors.
- Exponential current response allows translinear devices to be used as analog current multipliers:



$$I_i = I_0 e^{\alpha V_i}$$

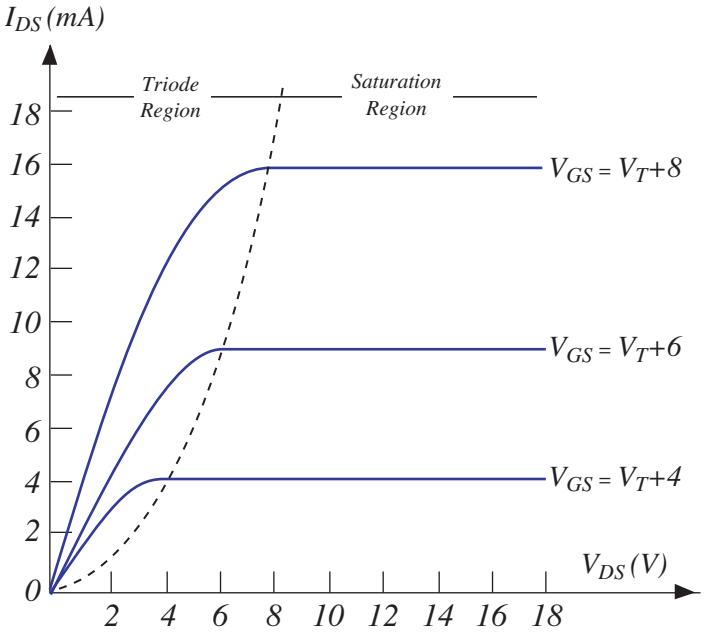
- Summing voltages around the loop gives $V_1 + V_4 = V_2 + V_3$. We can rewrite this as $\log(I_1) + \log(I_4) = \log(I_2) + \log(I_3)$, and thus:

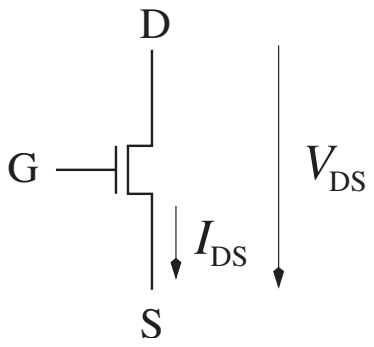
$$I_1 \, I_4 = I_2 \, I_3$$

Translinear Principle:
In a closed loop consisting of translinear devices with equal numbers of clockwise and counter-clockwise currents, the product of currents in the clockwise direction is equal to the product of currents in the counterclockwise direction.

Subthreshold MOS Model

A MOS transistor with a gate-to-source voltage V_{GS} lower than its threshold voltage V_{Th} has a very low drain current which responds exponentially to V_{GS}. It can therefore be operated as a translinear device.





$$I_{DS} = I_0 \frac{W}{L} \exp\left(\frac{\kappa(V_G - V_S)}{U_T}\right)\left[1 - \exp\left(\frac{-V_{DS}}{U_T}\right)\right]$$

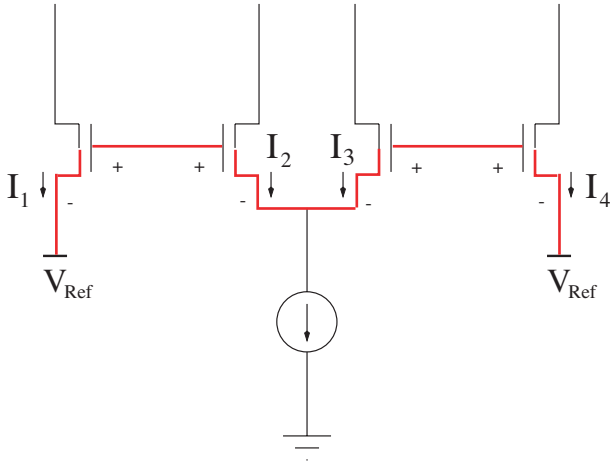
- where I_0 is a process constant, $\frac{W}{L}$ is the transistor's width-to-length ratio, $U_T \cong 26mV$, and $\kappa \approx 0.7$.
- If $V_{DS} > 100mV$, the transistor is said to be in saturation, and we may make the following approximation:

$$I_{DS} \cong I_0 \frac{W}{L} \exp\left(\frac{\kappa V_{GS}}{U_T}\right)$$

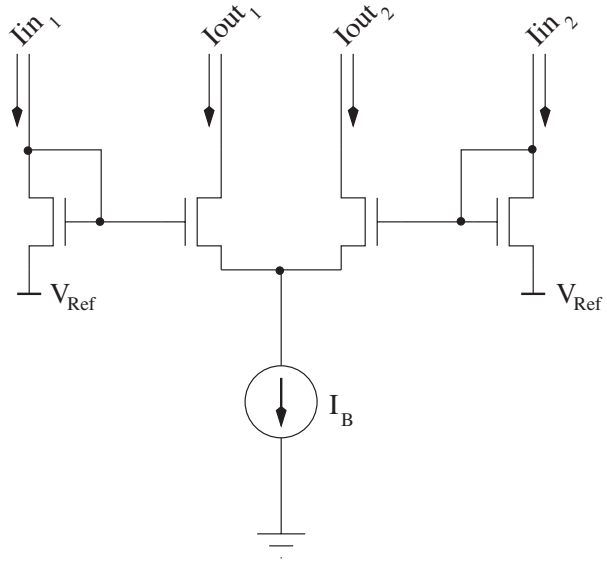
MOS Translinear Loop

The translinear principle may directly be applied to analyze networks such as

$$I_2\, I_4 = I_1\, I_3$$



- Following a loop from V_{Ref} to V_{Ref}, we find that I_2 and I_4 flow with the loop, while I_1 and I_3 flow against the loop. Therefore $I_2\, I_4 = I_1\, I_3$.
- The same analysis applies to the more realistic differential circuit:

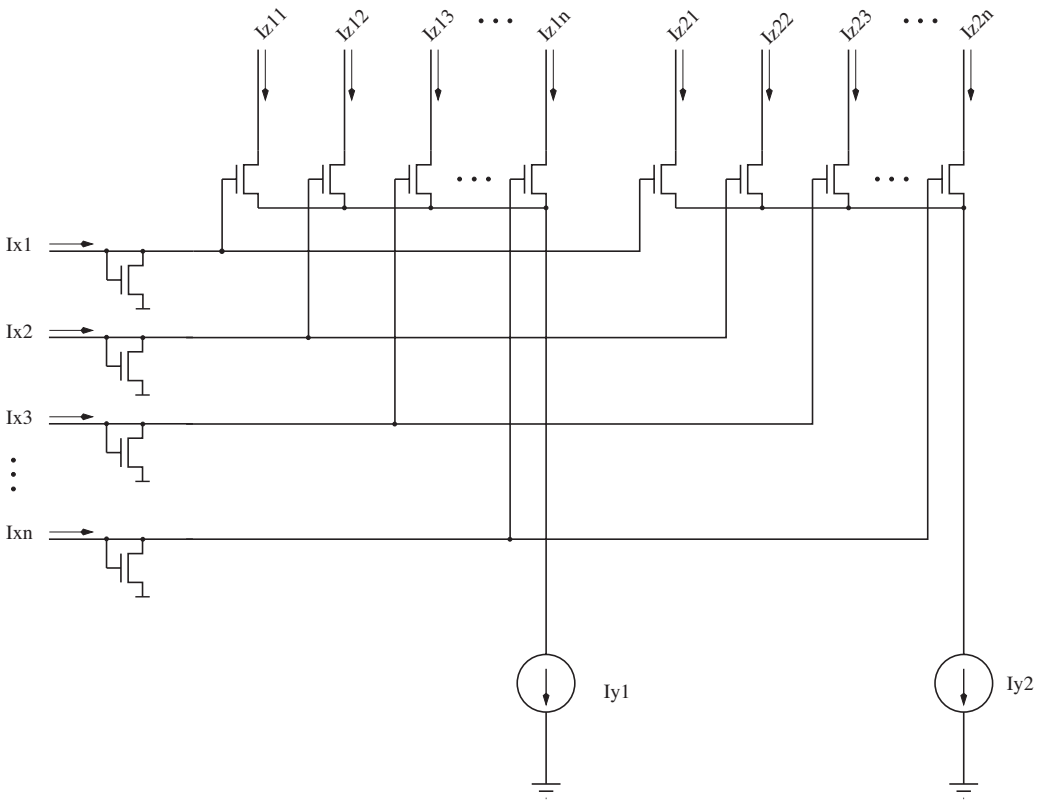


$$I_{\mathrm{out}_1} = \frac{I_{\mathrm{in}_1} \cdot I_B}{I_{\mathrm{in}_1} + I_{\mathrm{in}_2}}; \qquad I_{\mathrm{out}_2} = \frac{I_{\mathrm{in}_2} \cdot I_B}{I_{\mathrm{in}_1} + I_{\mathrm{in}_2}}$$

- This is the basis of the **Gilbert Multiplier** or **Vector Normalization** circuits.

Gilbert Multiplier

- The differential pair circuit may be expanded by adding more source-connected transistors. This arrangement is known as the Gilbert multiplier.



Let $I_{\text{tot}} = \sum_i Ix_i$. Then

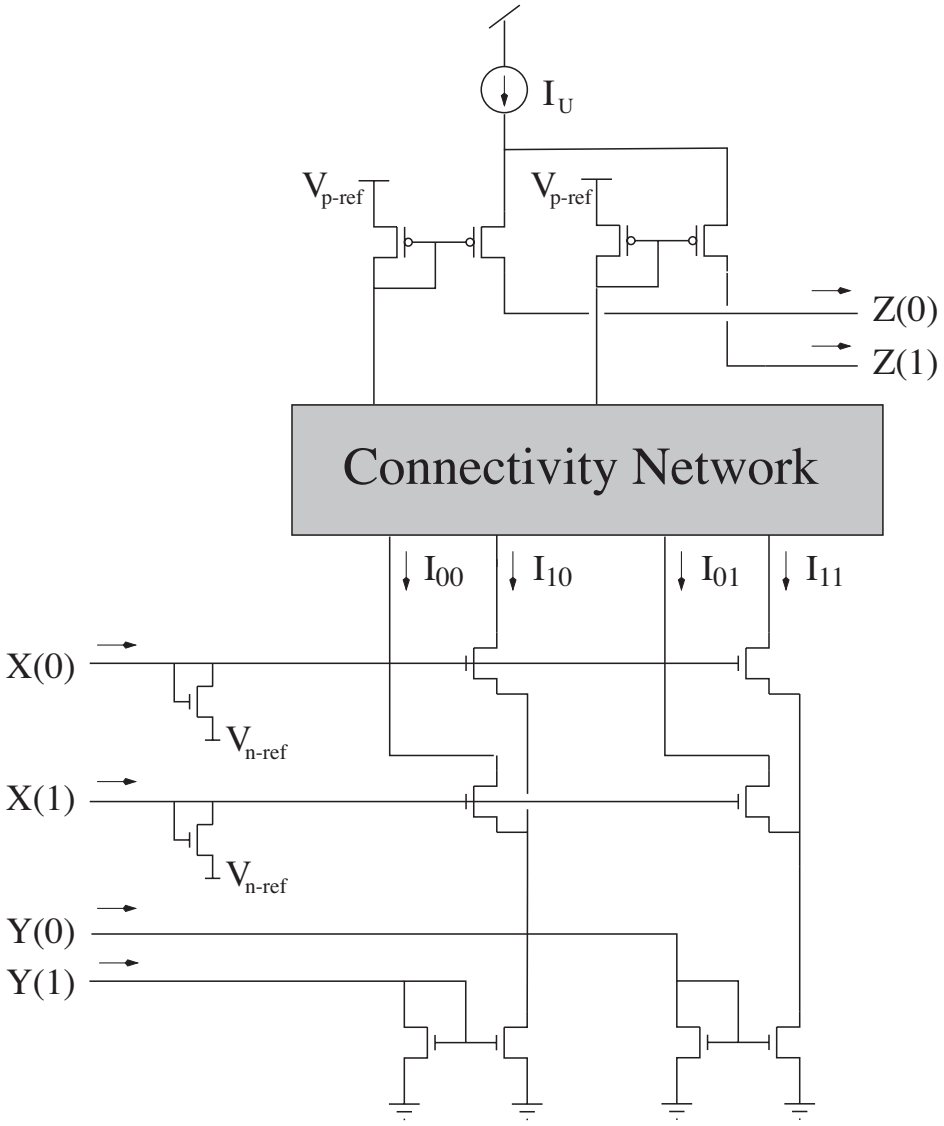
$$Iz_{ij} = \frac{Ix_i \cdot Iy_j}{I_{\text{tot}}}$$

Building Block

The Gilbert Cell forms the building block for vector normalization, current-mode sum, product, and normalization functions – everything needed for soft message passing decoding.

Basic Cell Structure

From the Gilbert Multiplier Cell a basic cell structure is derived



Multiplication The internal circuits $I_{ij} = X(i)Y(j)$ are **all possible** products of the input currents.

Addition Is performed by simply adding wires in the connectivity network to accomplish a given function.

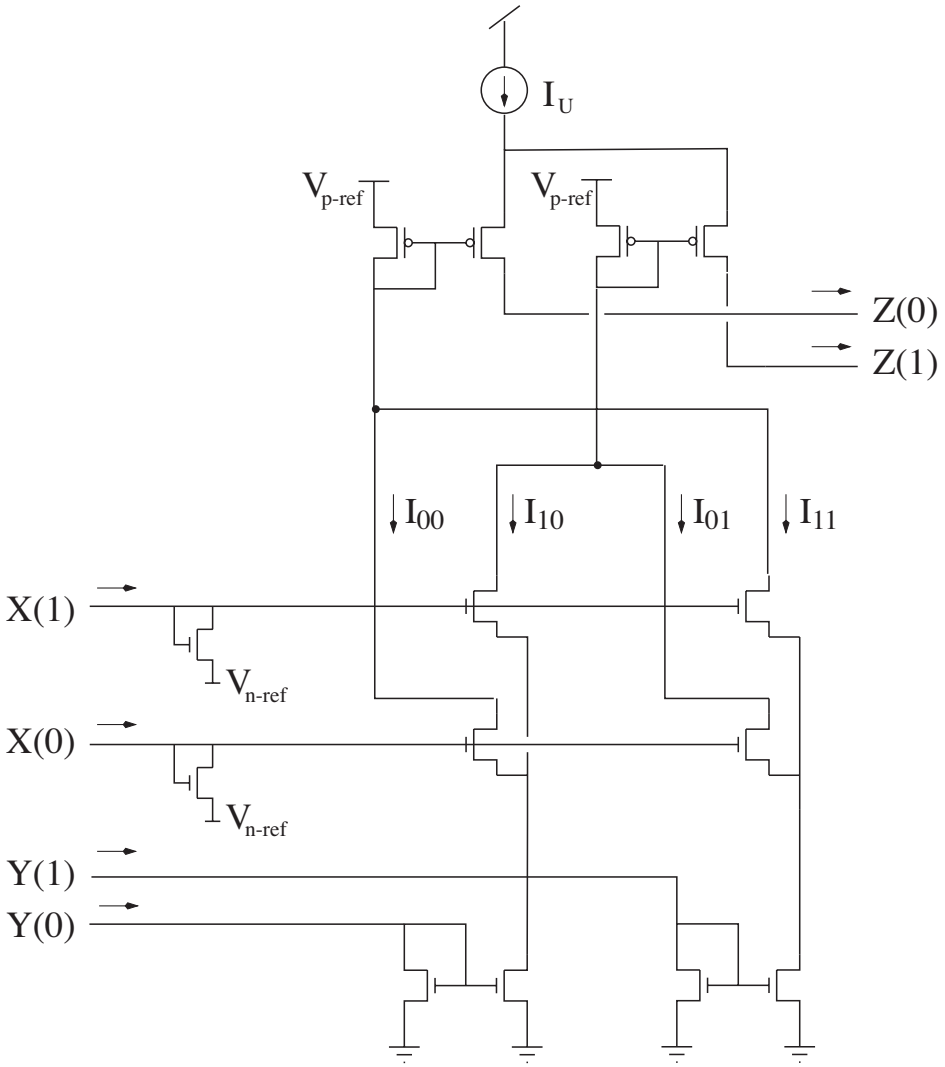
Output Stage The p-type current mirrors at the output reorient the currents to be used in another cell as input.

Example: Check Node Circuit

A parity-check node needs to compute at its outputs:

$$\begin{aligned} Z(0) &= X(0)Y(0) + X(1)Y(1) \\ Z(1) &= X(1)Y(0) + X(0)Y(1) \end{aligned}$$

This is accomplished with the following circuit:



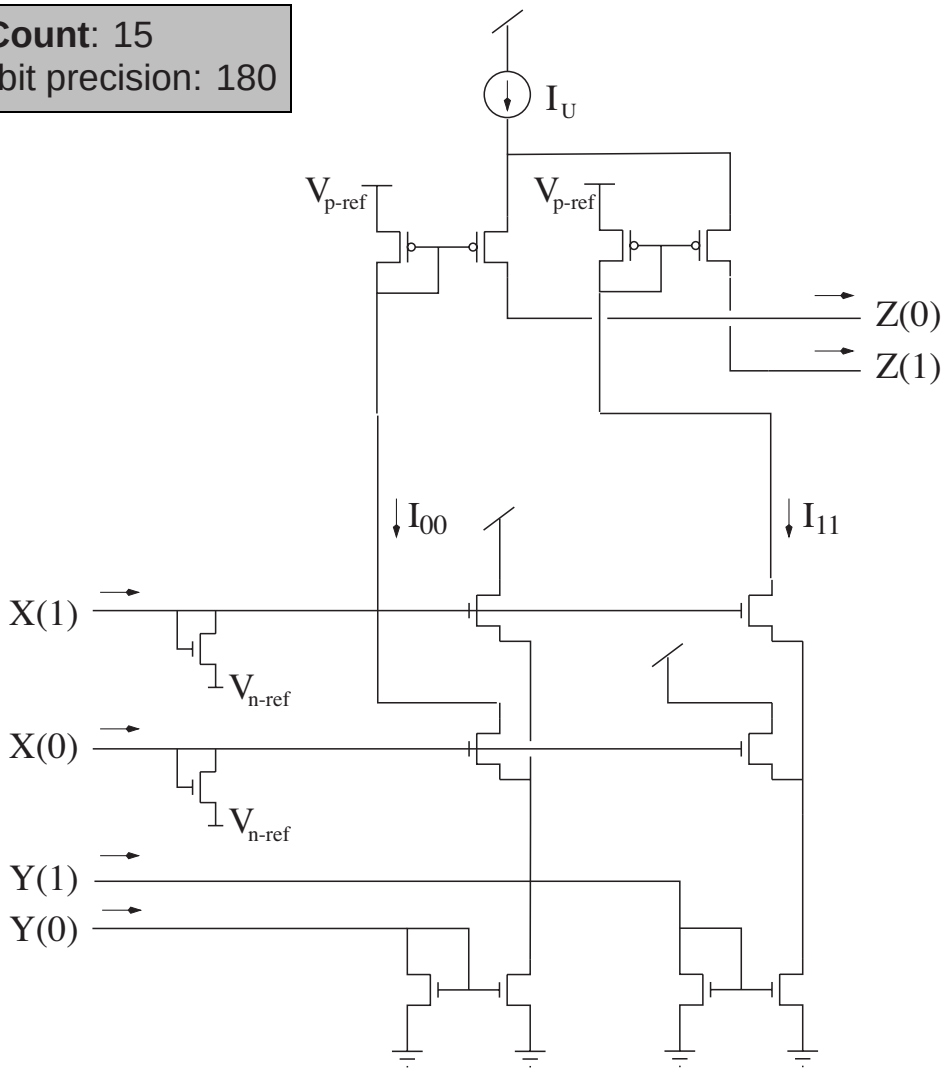
Example: Equality Node Circuit

The equality node (variable node in an LDPC) needs to compute:

$$\begin{aligned} Z(0) &= \propto X(0)Y(0) \\ Z(1) &= \propto X(1)Y(1) \end{aligned}$$

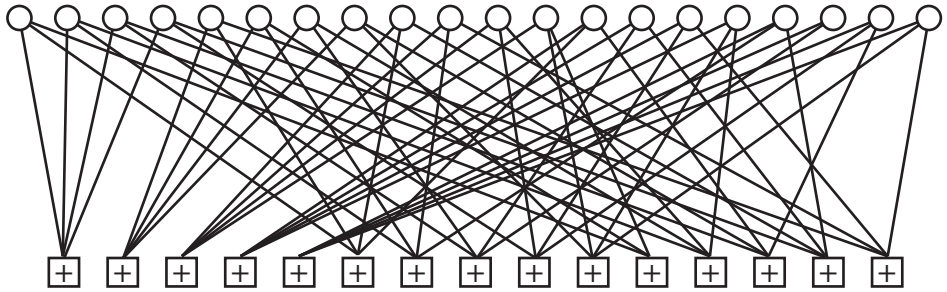
This is accomplished with the following circuit:

Transistor Count: 15
(Analog, 10-bit precision: 180



Analog Decoding: Promise

With these simple elements, an LDPC decoder (and others) can be bulit:

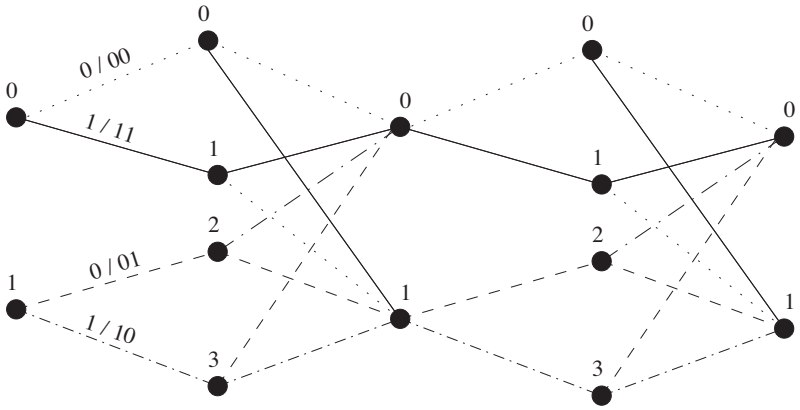


Advantages:

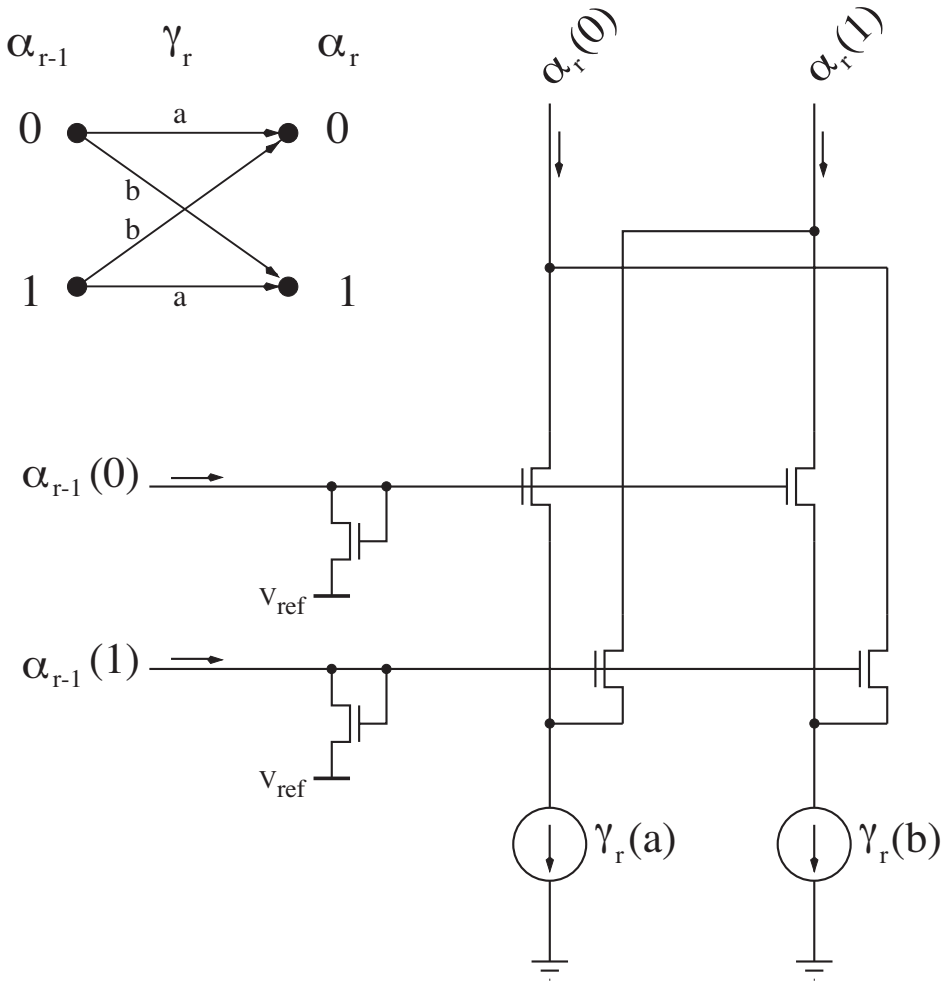
- The MOS transistor is biased in the **weak-inversion** or **subthreshold** region, where is consumes typically less than 100nA.
- The transistor is never turned “on” and operates with “leakage current”
- The power consumed is in the nano-Watt range
- The transistor is slow – throughput is achieved through massive parallelism Large codes can achieve throughputs in excess of 1Gb/s.
- CMOS technology can be used, which has many advantages, such as cheap fabrication and small transistor sizes.
- CMOS are well-suited for systems-on-a-chip ASICS
- The analog decoder produces no high-frequency interference

Example – [8,4] Extended Hamming Code

The [8,4] Extended Hamming code has the following tailbiting trellis:



This code is decoded via APP decoding, which can use the same analog building blocks. The code's fundamental **Butterfly** structure has the following simple implementation:

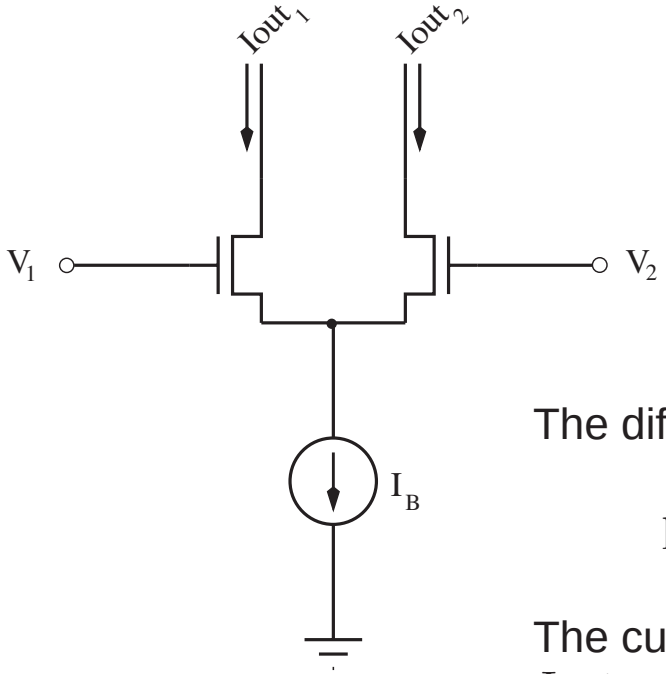


Input Stages:

The input signals are received as **voltage values** which need to be converted to probability values. Since

$$\mathrm{LLR} = \log\left(\frac{\Pr(x=1|y)}{\Pr(x=0|y)}\right) = \frac{4y}{N_0}$$

we need to convert the input signal y which appears as a voltage into proportional probability currents. This is done by a differential input stage:



The differential stage generates:

$$\log\left(\frac{I\mathrm{out}_1}{I\mathrm{out}_0}\right) = \propto (V_1 - V_2)$$

The currents are normalized to $I\mathrm{out}_1 + I\mathrm{out}_0 = I_\mathrm{B}$, which represents unit probability.

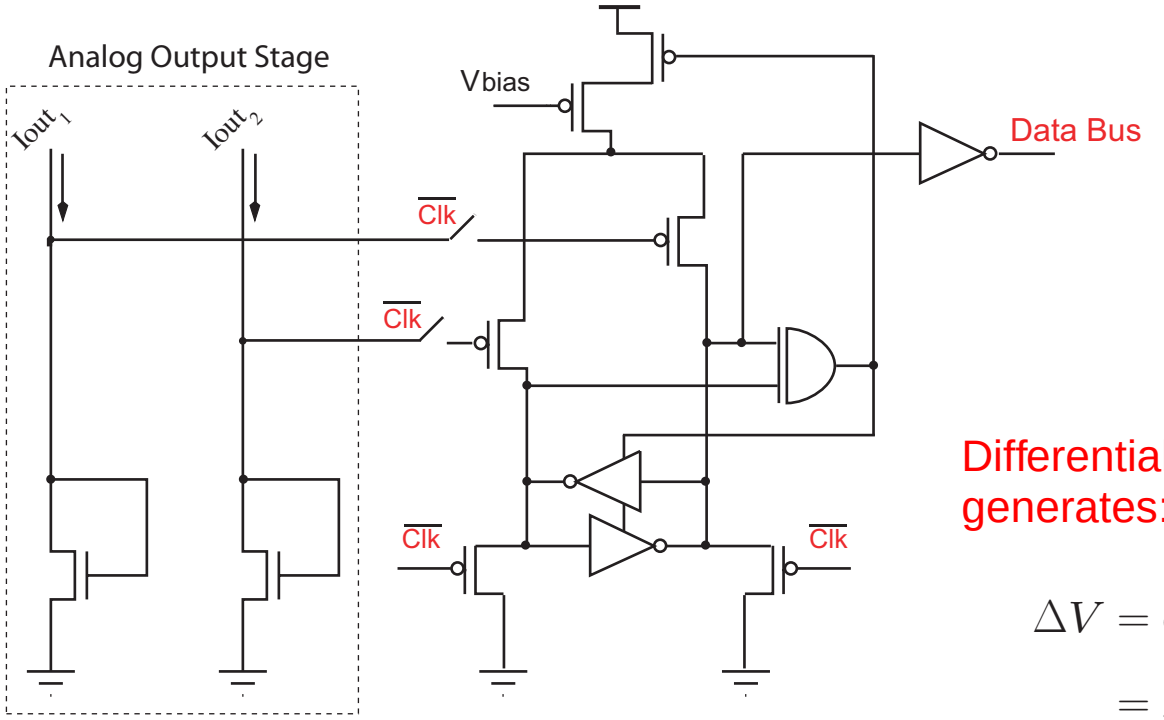
Serial Interface

A Serial Interface is used to move serial channel samples into a sample-and-hold chain whose outputs are presented to the decoder in parallel.

[Win04]	C. Winstead, J. Die, S. Yu, C. Myers, R. Harrison, and C. Schlegel, "CMOS Analog MAP decoder for an (8,4) Hamming code," *IEEE J. Solid State Cir.*, Vol. 29, No. 1, pp. 122–131, January 2004.

Output Stages

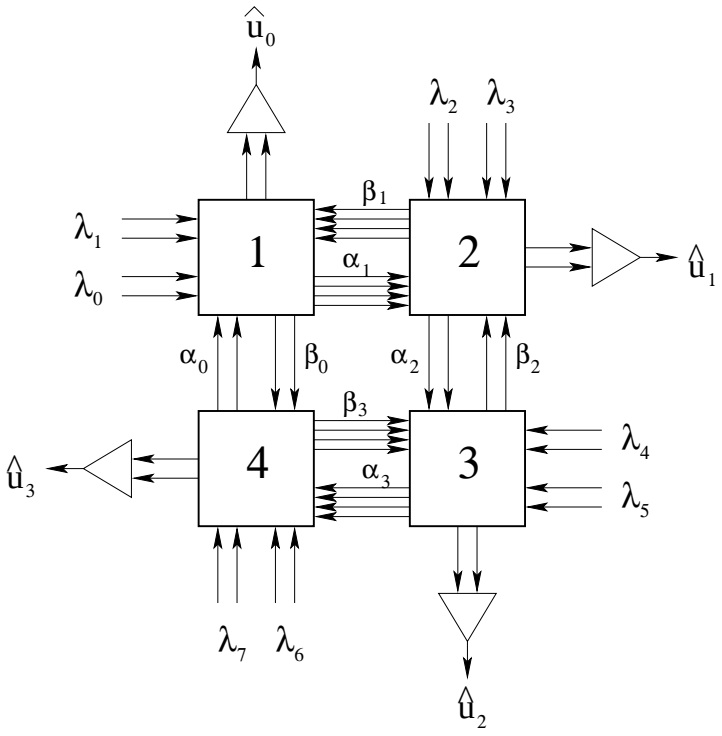
At the output, the signal needs to be **converted back to voltages** which are being fed into conventional comparator circuits.



Differential stage generates:

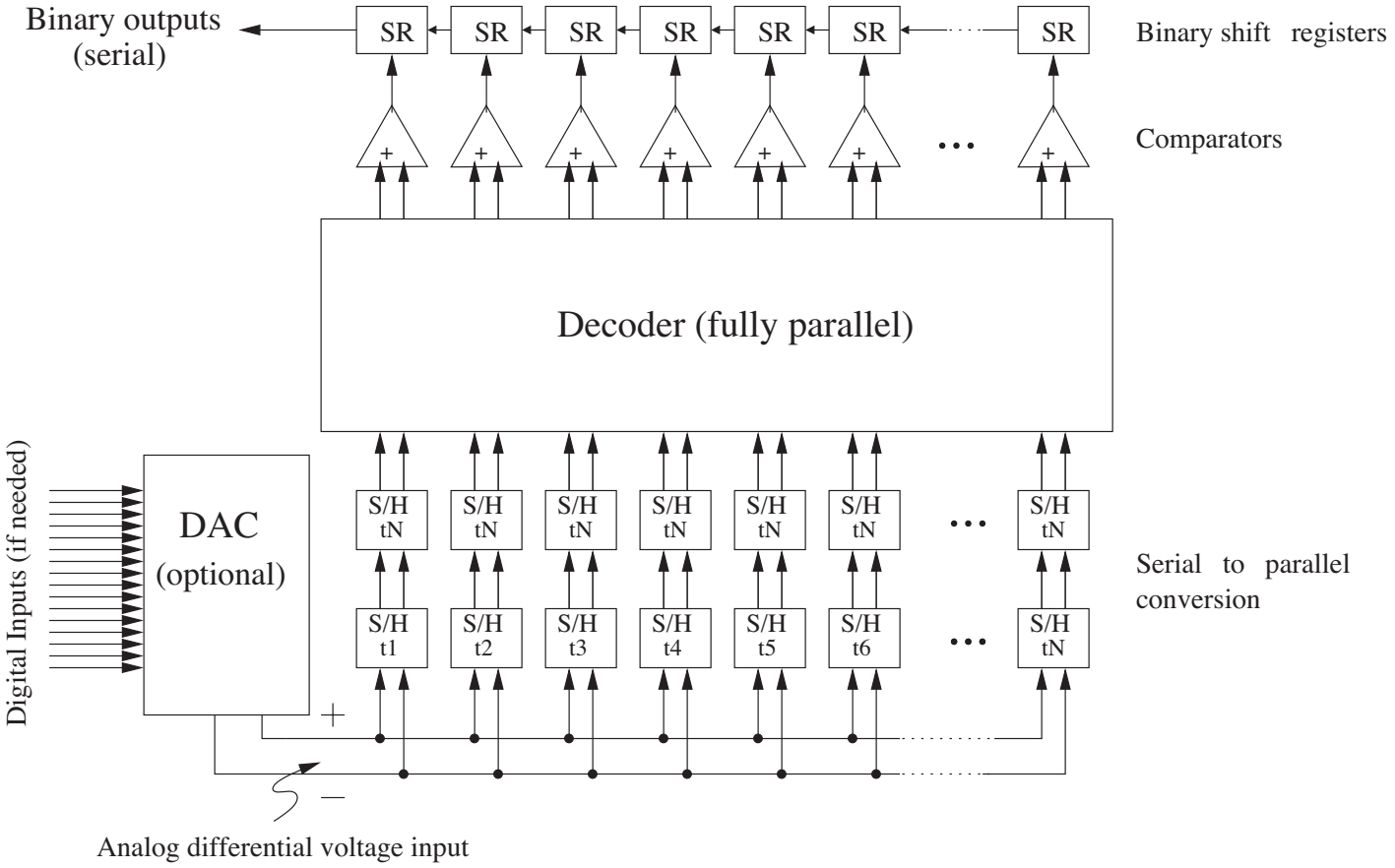
$$\Delta V = \propto \exp\left(\frac{I\text{out}_1}{I\text{out}_0}\right) = \lambda$$

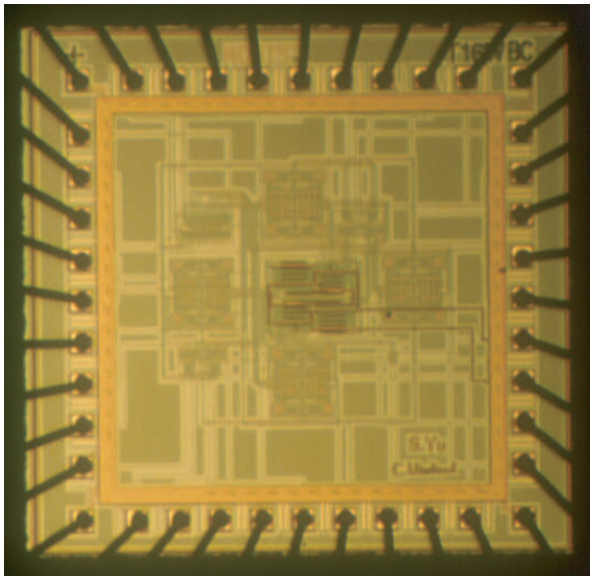
The complete Hamming decoder then has the blocks layout:



Complete Decoder

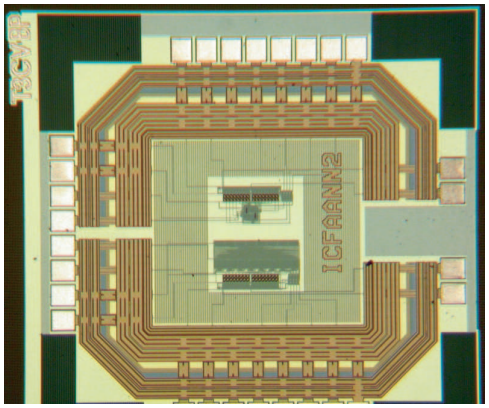
The complete decoder comprises a differential analog input line, serial-to-parallel conversion, and a parallel-to-serial output line:



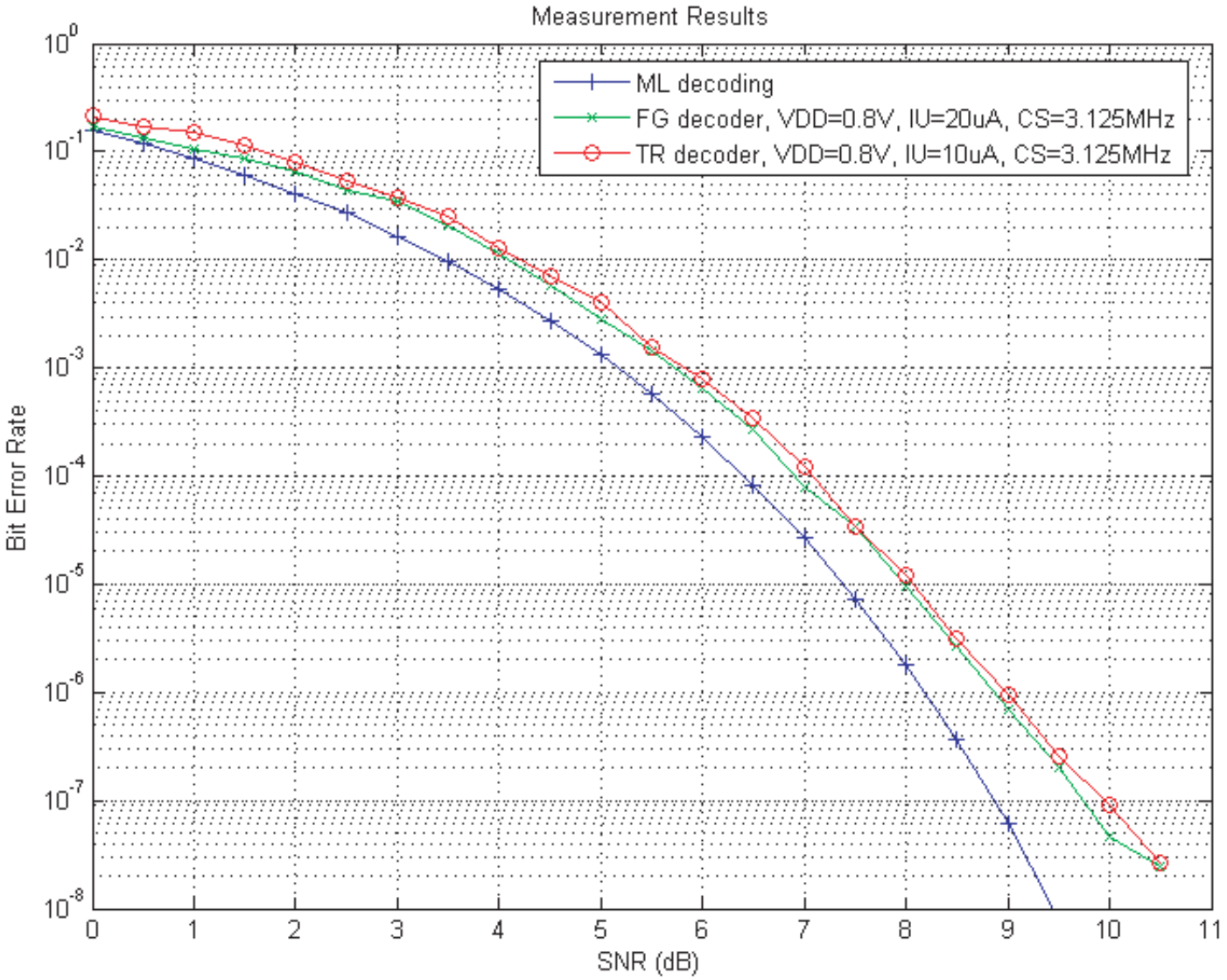


- Fabricated in AMI 0.5 micron process
- Die size is 1.5mm by 1.5 mm
- Fabricated through Canadian Microcorporation (CMC) University program

Does It Work?



- Fabricated in TSMC 0.18 micron process
- Fabricated through Canadian Microcorporation (CMC) University program

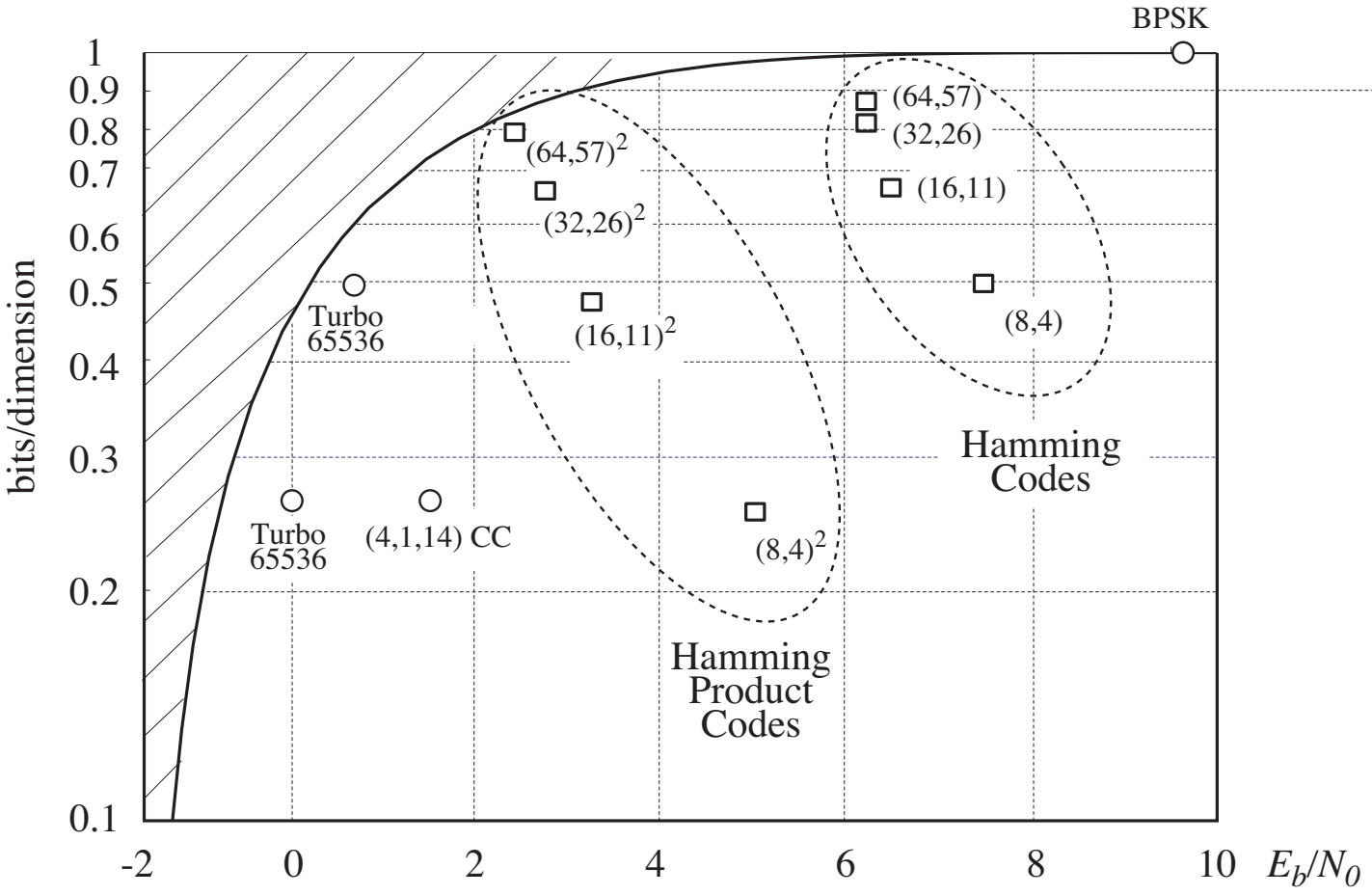


Codes of Practical Size

For practical communication systems larger codes with larger gain are required. We built a $(16, 11)^2$ product code.

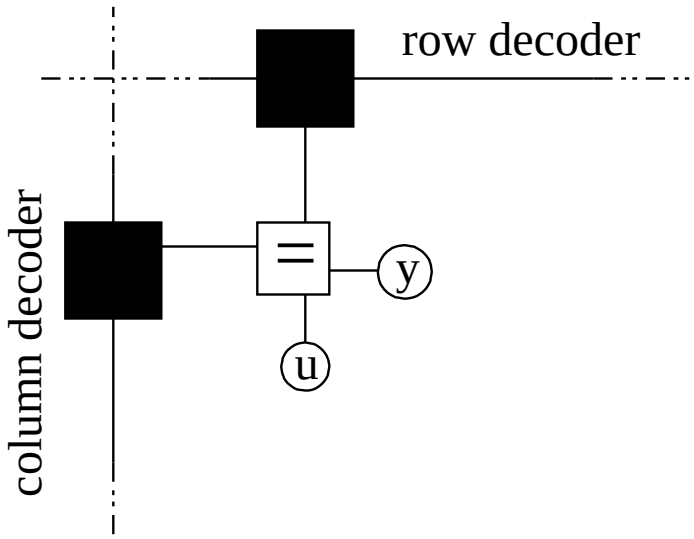
While these codes with iterative decoding are not fully competitive with turbo or LDPC codes, they do possess some advantages:

- Easy codeword geometry which allows finding $d_{\min}$ and nearest neighbors easily
- Small numbers of iterations to achieve limit performance
- Small core sizes for high-speed implementations – see AHA core products

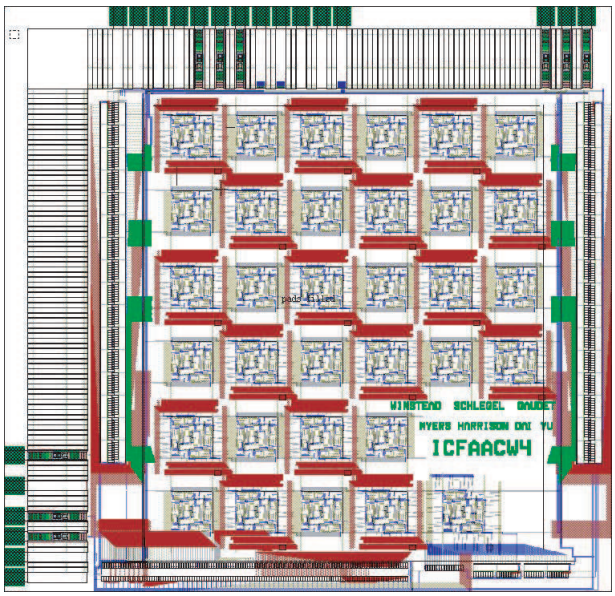


Decoder Architecture

The decoder architecture is a mixture of an analog trellis decoder like the [8,4] Hamming, and an LDPC code:



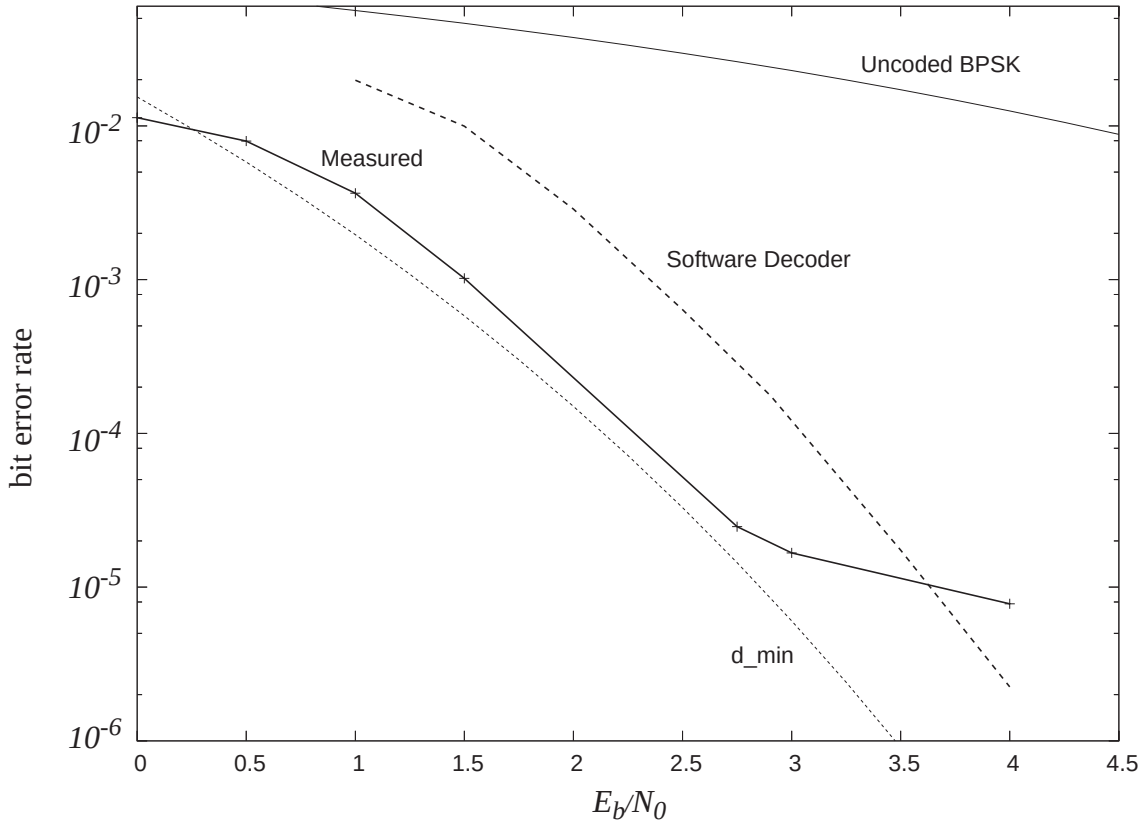
Rows and Columns are decoded via a trellis decoder, and bits that are shared are connected with an equality node. The structure of the decoder can be seen on the chip layout:



- Built in TSMC 0.18 micron process
- Die size is 2.3mm by 2.5mm
- Fabricated through CMC's University program

Performance Measurement Results

The product decoder chip is currently undergoing extensive testing.



- **Software Decoder**
  This is a simulation result using the iterative digital decoding algorithm discussed earlier
- $d_{\min}$ **Curve**
  This is an approximation of the optimial decoder performance given by

$$P_b \approx N_{d_{\min}} Q\left(\frac{d_{\min}}{\sqrt{2N_0}}\right)$$

  where $N_{d_{\min}}$ is the number of codewords that have a given bit in error and are at a distance $d_{\min}$ from the transmitted codeword
- **Measurements** are actual measured BERs on a single bit of the analog product decoder

Simulation and Verification

Monte Carlo Simulations

The decoder chain is simulated and a number N_{MC} of instances x_i of the process are run in order to obtain an estimate of the error probability P:

$$\tilde{P} \approx \frac{1}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} I(x_i)$$

where $I(x_i) = 1$ if there is an error.

Reliability of Simulation
The expected value of P is

$$\mathrm{E}\big[\tilde{P}\big] = \frac{1}{N_{\mathrm{MC}}} N_{\mathrm{MC}}\, \mathrm{E}[I(x_i)] = P$$

That is, $\tilde{P}$ is an **unbiased estimator** of P.

Variance of the Estimation

$$\begin{aligned}
\sigma^2_{\mathrm{MC}} &= \mathrm{E}\big[\tilde{P}^2\big] - \mathrm{E}\big[\tilde{P}\big]^2 \\
&= \frac{1}{N^2_{\mathrm{MC}}}\, \mathrm{E}\left[\sum_{i=1}^{N_{\mathrm{MC}}} I(x_i) \sum_{j=1}^{N_{\mathrm{MC}}} I(x_j)\right] - P^2 \\
&= \frac{P}{N_{\mathrm{MC}}} + \frac{N_{\mathrm{MC}} - 1}{N_{\mathrm{MC}}} P^2 - P^2 = \frac{P(1-P)}{N_{\mathrm{MC}}}
\end{aligned}$$

The variance, in turn, can be estimated as

$$\tilde{\sigma}^2_{\mathrm{MC}} = \frac{\frac{1}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} I(x_i) - \left(\frac{1}{N_{\mathrm{MC}}} \sum_{i=1}^{N_{\mathrm{MC}}} I(x_i)\right)^2}{N_{\mathrm{MC}}}$$

which is **an unbiased estimator** for the variance σ^2_{MC}.

Evaluating Decoder Performance – Importance Sampling

- Software simulation of very low error rates are usually not feasible with Monte-Carlo Simulation.
- Under certain circumstances, an accelerated technique **Importance Sampling** can be used.

Formulation

This issue is one of finding an integral of the general form

$$y = \int_{\Omega} f(x)dx; \quad \Omega \text{ is the integration domain}$$

IS evaluates this integral as

$$\int_{\Omega} f(x)dx = \int_{\Omega} \frac{f(x)}{\rho(x)}\rho(x) = \int_{\Omega} w(x)\rho(x)dx$$

The weighting function $w(x) = f(x)/\rho(x)$ changes the distribution of the samples over Ω.

- Using finite point approximations, we have

$$y \approx \frac{1}{N_s}\sum_{i=1}^{N_s} w(x_i)$$

  where the new random samples are drawn according to $\rho(x)$.
- It can be shown that the optimal weighting function using

$$\rho_{\text{opt}}(x) = \frac{|f(x)|}{\int_{\Omega}|f(x)|dx}; \quad x \in \Omega$$

  leads to a constant weighting function $w(x) = \int_{\Omega}|f(x)|dx$ – which would require only a single sample.

Importance Sampling

Using importance sampling, the error is estimated as

$$\tilde{P} = \frac{1}{N_s} \sum_{i=1}^{N_s} w(x_i)$$

with variance:

$$\begin{aligned}
\sigma_s^2 &= \frac{1}{N_s^2} \mathrm{E}\left[\sum_{i=1}^{N_s} w(x_i) \sum_{j=1}^{N_s} w(x_j) \right] - P^2 \\
&= \frac{1}{N_s} \mathrm{E}\left[\sum_{i=1}^{N_s} w^2(x_i) \right] + \frac{N_s - 1}{N_s} P^2 - P^2 \\
&= \frac{1}{N_s} \mathrm{E}\left[\sum_{i=1}^{N_s} w^2(x_i) \right] - \frac{P}{N_s}
\end{aligned}$$

An unbiased estimator for the variance is given by

$$\tilde{\sigma}_{\mathrm{IS}}^2 = \frac{1}{N_s^2} \sum_{i=1}^{N_s} w^2(x_i) - \frac{1}{N_s} \left(\frac{1}{N_s} \sum_{i=1}^{N_s} w(x_i) \right)^2$$

Gain: The gain of IS versus Monte-Carlo is expressed as

$$G_{\mathrm{IS}} = \frac{\sigma_{\mathrm{MC}}^2}{\sigma_{\mathrm{IS}}^2}$$

The key is to ensure that the gain $G_{\mathrm{IS}} > 1$ order to save on the number of simulation runs.

Gain of Importance Sampling

Note that

$$\frac{1}{N_s}\sum_{i=1}^{N_s} w^2(x_i) - \left(\frac{1}{N_s}\sum_{i=1}^{N_s} w(x_i)\right)^2 \geq 0$$

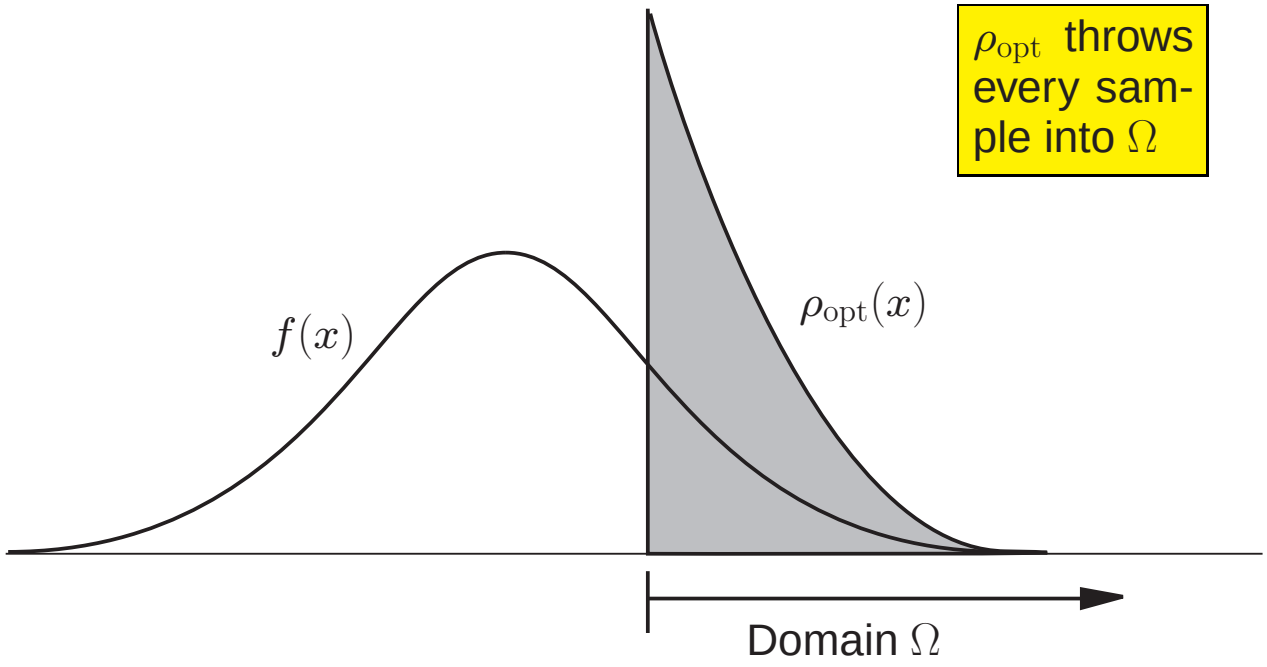
due to **Jensen's Inequality**, with equality if and only iff $w(x_i)$ is a constant.

If we set

$$w(x) = \int_\Omega |f(x)|dx = \text{constant}$$

the variance $\tilde{\sigma}^2_{\mathrm{IS}}$ goes to zero.

The related shifted probability density function $\rho_{\mathrm{opt}}(x)$ moves probability mass into the area of integration, and biases the count. Ideally, all mass is moved into the area of interest.

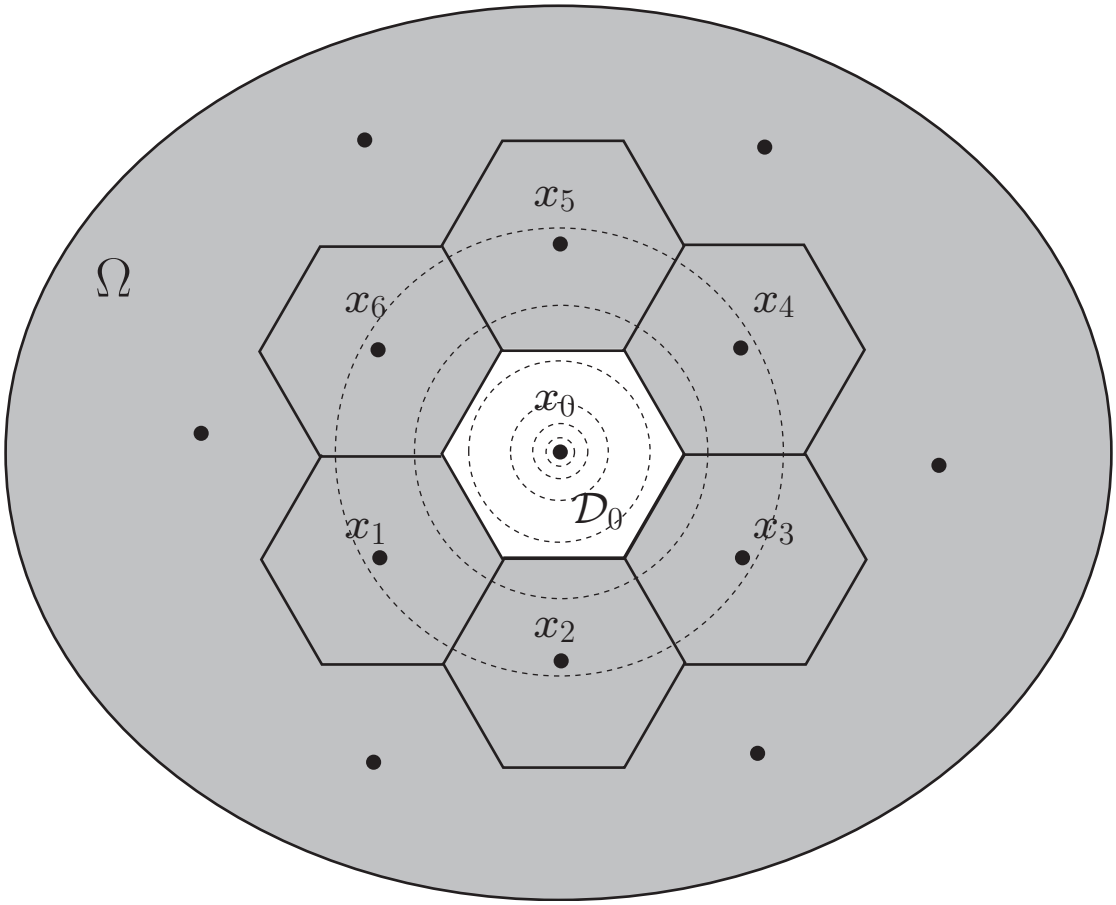


Application to FEC Performance Evaluation

Error Probability of Codeword x_0

P_0 is obtained by integrating the conditional channel pdf $p(y|x_0)$ over the complement of the decision region $\mathcal{D}_0$ of x_0.

$$P_0 = \int_{\cup \mathcal{D}_i; i \neq 0} p(y|x_0) dy = \sum_{i=1}^{M} \int_{\mathcal{D}_i} p(y|x_0) dy$$



P_0 can be approximated by concentrating on the most probable error neighborhoods by restricting the explored error neighborhoods to those in the immediate proximity of x_0.

Importance Sampling via Mean Translation

In general, we try to bias the noise towards producing more errors. This can be accomplished in a number of ways:

- **Excision** – certain samples are recognized as not causing an error and can be discarded without simulating. E.g., if simple slicing causes all of the bits to be correct, the decoder will complete successfully.
- **Variance Scaling** – the noise variance is simply increased and thus causes more errors. Since the weight function

$$w(y) = \frac{\sigma_B}{\sigma} \exp\left(-|y - x_0|^2 \frac{\sigma_B^2 - \sigma^2}{\sigma_B^2 \sigma^2}\right) \approx \exp\left(-|y - x_0|^2/\sigma^2\right)$$

  is exponential in the SNR, variance scaling does not work well.
- **Mean Translation** – samples are generated according to $p^*(y) = p(y-\mu)$, where μ is a shift value towards the **decision boundary**. We get

$$\begin{aligned} P_{i0} &= \int_{\mathcal{D}_i} \frac{p(y|x_0)}{p(y-\mu|x_0)} p(y-\mu|x_0)dy \\ \Longrightarrow P_{i0} &\approx \frac{1}{N_s} \sum_{j=1}^{N_s} \frac{p(y|x_0)}{p(y_j-\mu|x_0)} p(y_j-\mu|x_0) I(y_i) \end{aligned}$$

The most successful way of performing IS has been via a simple translation of the mean. Typical shifts are to the (approximate) decision boundary

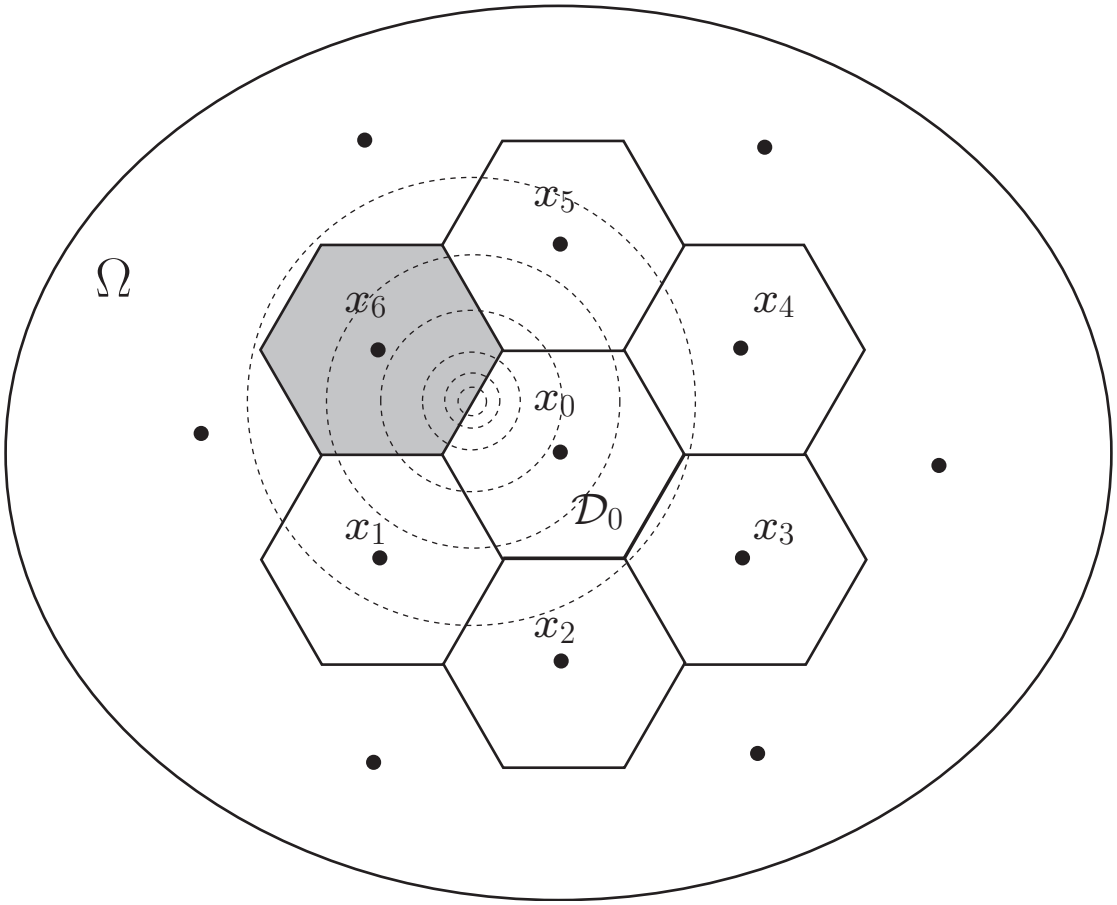
$$\mu = \frac{x_0 + x_i}{2}$$

Error Probability via IS

If the codeword structure of the immediate neighborhood is well known, we can successively bias towards each error codeword and sum up the error rates to obtain the estimate:

$$P_0 = \sum_{i=1}^{M'} P_{0i}; \quad M' \leq M$$

where P_{0i} is calculated via IS and biasing to $\mu = (x_i - x_0)/2$:



Gain of IS

Monte-Carlo
the variance of Monte-Carlo simulations is

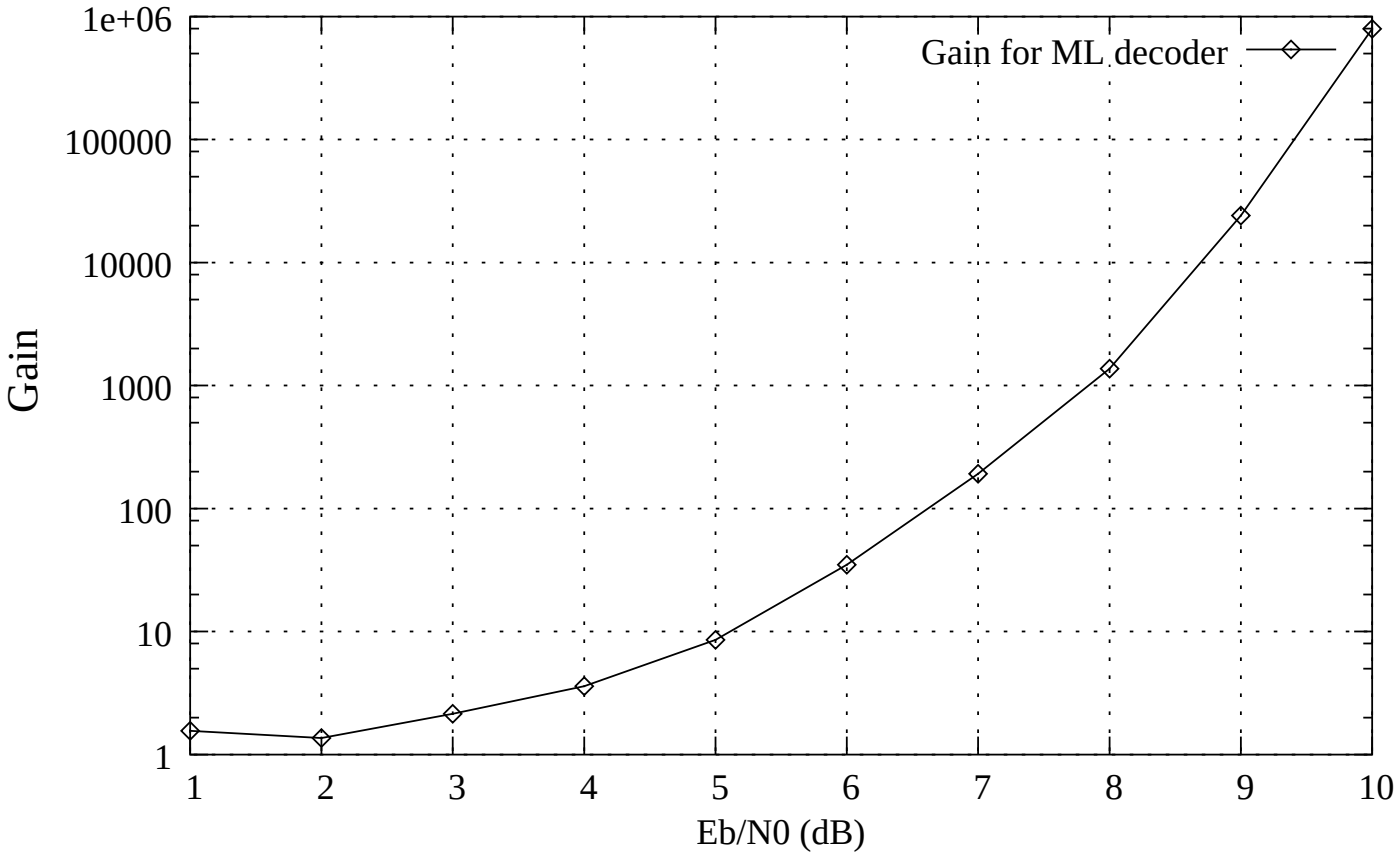
$$\mathrm{var}(P_0) = \frac{P_0(1-P_0)}{N_{\mathrm{MC}}}$$

Importance Sampling
The variance of the IS technique is

$$\mathrm{var}(P_0) = \frac{1}{N_s}\sum_{j=1}^{N_s} I(y_i) - P_0^2$$

Gain Example: The ratio of the number of samples to achieve the same variance is the gain. The gain of IS over Monte-Carlo can be astronomical:

IS Simulation Gain of a (7,4) Hamming code

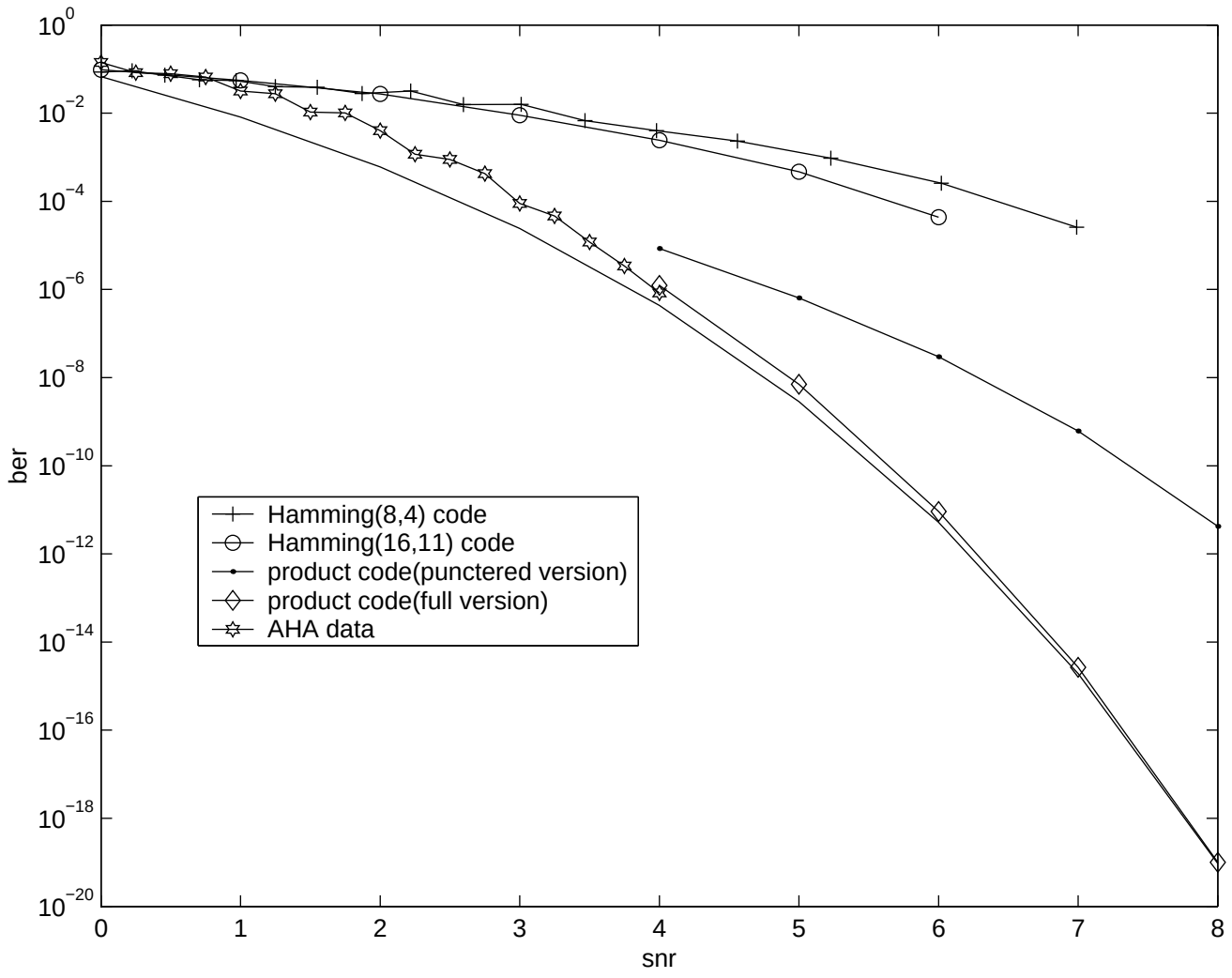


Application to the Product Decoder

Note: For a single bit, only the bit neighbors need to be considered, using just the minimum-distance codeword, extremely low error rates can be simulated.

Note: IS can be effective if the decoder is not maximum likelihood, and the conventional union bound is not appropriate.

Simulation results using IS:



[Dai01] J. Dai, *Design Methodology for Analog VLSI Implementations of Error Control Decoders*, PhD thesis, University of Utah, 2001.

Effects of Physics

One has to carefully address physical effects that could have influence on the behavior of the code. The prominent such effects are:

- **Device Mismatch**
  the circuit relies on a multitude of current mirrors, these can only be build within a certain tolerance.

- **Comparator Offset Errors**
  Comparator exhibit undesired random offset voltages. The issue is largely one of comparator yield, i.e., what is the probability that all the comparators on a given circuit are functional.

- **Substrate Leakage Currents**
  affect “life” of the sampled signals. Stored voltages leak through the substrate, whereby the leakage currents are nearly constant – hence differential storage.

  Strong leakage also affects the computational units’ accuracy.

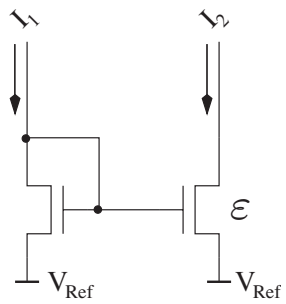
- **Channel Leakage Currents**
  make it difficult to mirror small currents due to large source-drain voltages across the mirror transistor.

- **Charge Injection**
  The S/H inject residual charge onto the storage capacitor when switches are opened. This has the effect of scaling the differential voltages.

Mismatch Effects in the Core

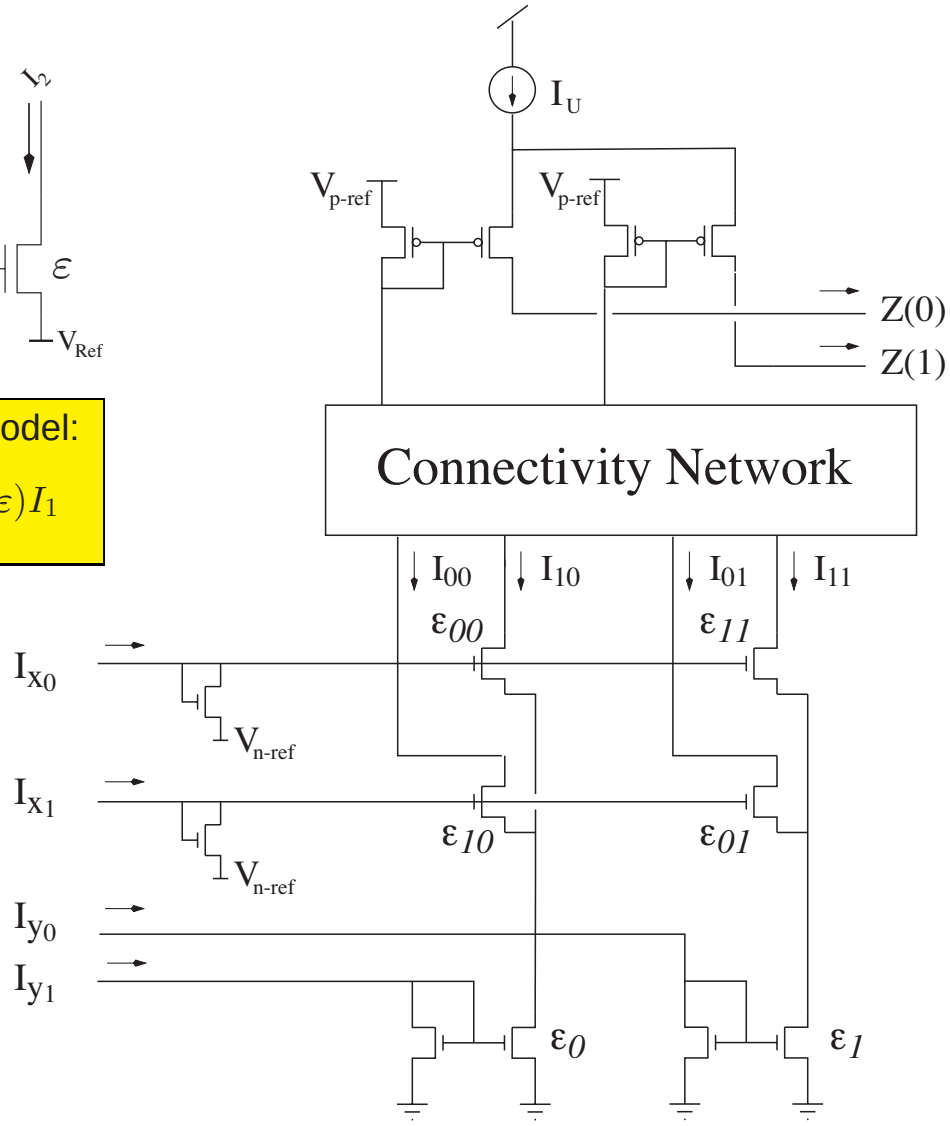
Probably the most disturbing issue is the one of transistor mismatch in large decoder circuits: Are they going to function properly?

Assume the following mismatch model where the mismatch parameters ε are assumed to be Gaussian distributed.



Mismatch Model:

$$I_2 = (1+\varepsilon)I_1$$



We can calculate the actual output currents as

$$I_{ij} = f(x, y, \varepsilon) = \frac{I_{x_0} I_{y_0} (1+\varepsilon_j)(1+\varepsilon_{\bar{i}\bar{j}})}{I_{x_i}(1+\varepsilon_{\bar{i}\bar{j}}) + I_{x_{\bar{i}}}(1+\varepsilon_{\bar{i}\bar{j}})}$$

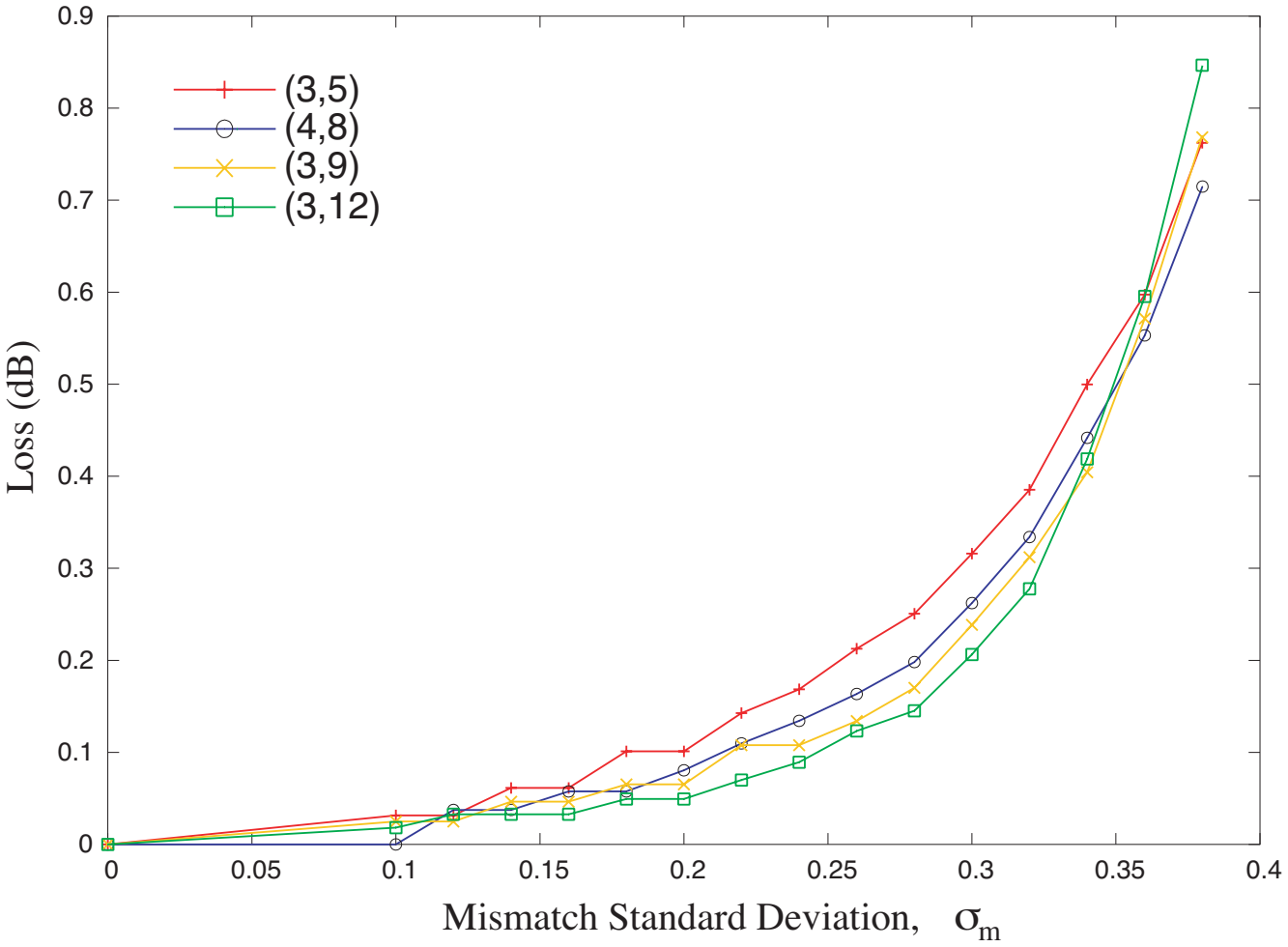
Density Evolution Analysis

We now assume the inputs are Gaussian distributed with mean and variances μ_x, μ_y and σ_x^2, σ_y^2. The output mean can now be calculated via numerical integration as:

$$\mu_z = \int f(x, y, \varepsilon) p_G(x) p_G(y) p_G(\varepsilon) dx dy d\varepsilon$$

The basic functions are then put together to build the node processors for an LDPC code and the thresholds are computed. The figure below plots the loss function

$$f_{\text{loss}}(\sigma_\varepsilon) = \frac{s^*(\sigma_\varepsilon)}{s^*(0)}$$



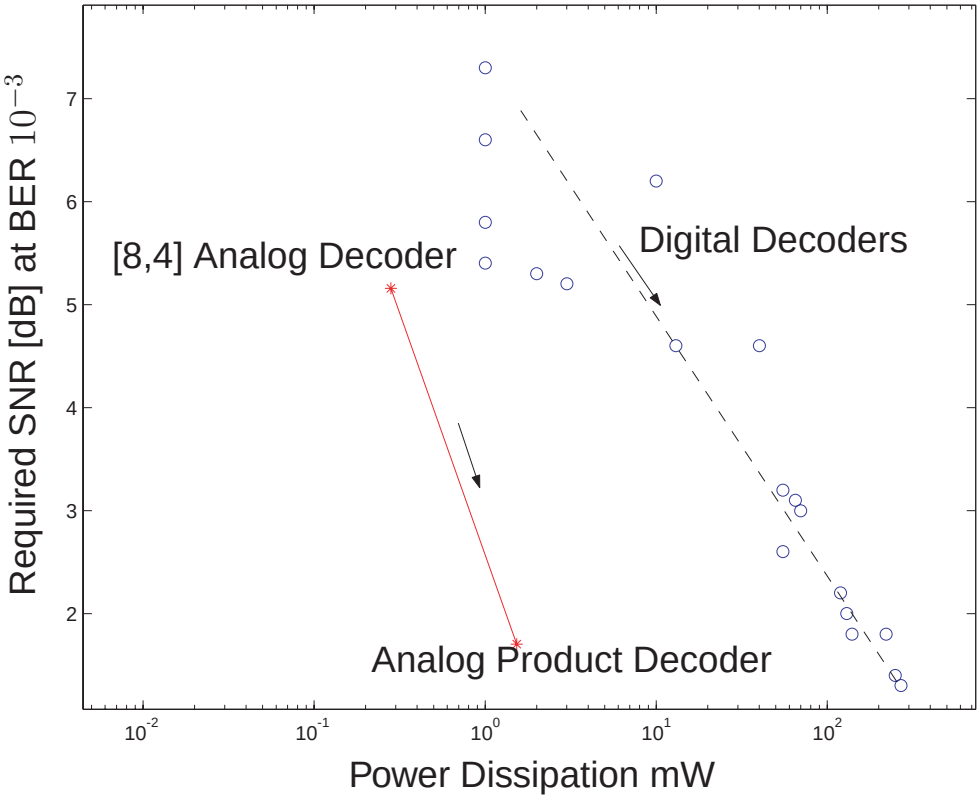
[Win04] C. Winstead, *Analog Iterative Error Control Decoders*, PhD Thesis, University of Alberta, 2004.

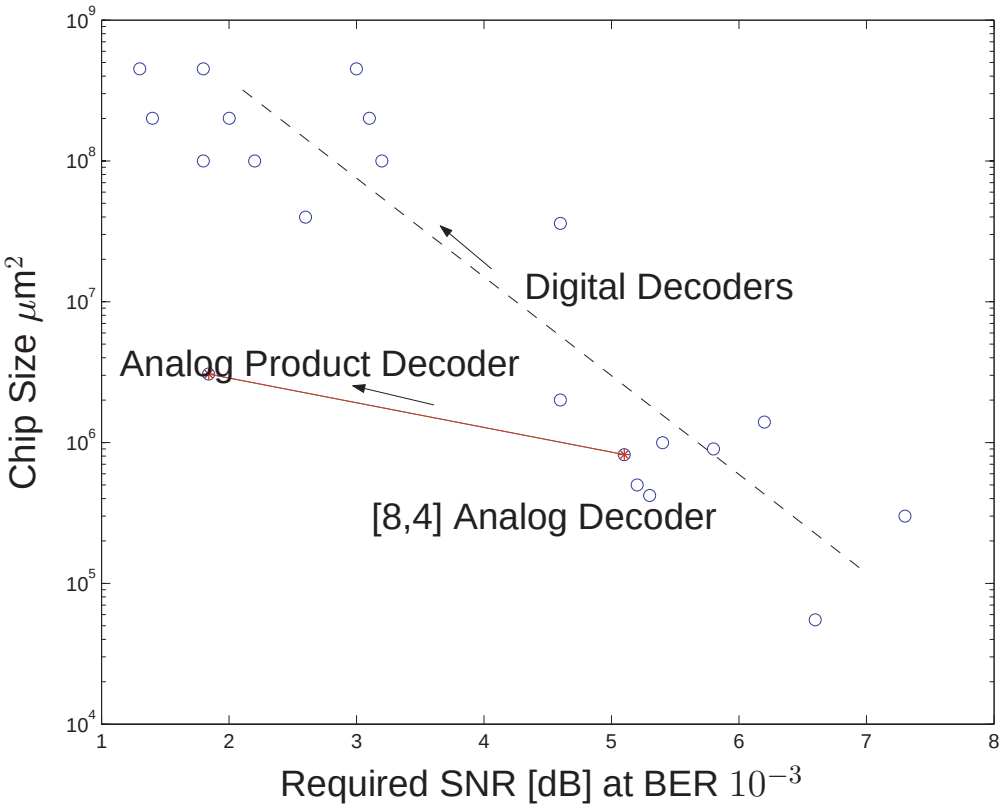
Some Recently Fabricated Analog Chips

Power efficiency

Who or What	Technology	Power	Throughput (info bits)	Energy / decoded bit
Gaudet et.al., JSCC'03	0.35μm	185mW @3.3V	13.3Mbps	14 nJ/b core,IO
Winstead et.al., JSCC'04	0.5μm	45mW @3.3V	1Mbps	45 nJ/b core,IO
Blanskby et.al., JSCC'02	0.16μm	690mW @1.8V	500Mbps	1.25 nJ/b (digital)
Bickerstaff, JSCC'02	0.18μm	290mW @1.8V	2Mbps	142 nJ/b (digital)
Moerz et. al., ISSCC'00	0.25μm	20mW @3.3V	160Mbps	0.13 nJ/b core
Vorig et. al., JSSC'05	0.35μm	10mW @3.3V	2Mbps	12.6 nJ/b core,IO
(16,11) ^{2} decoder	0.18 μm	7mW @1.8V	100 Mbps	0.07 nJ/b core, IO
Factor g raph decoder	0.18 μm	.283mW @1.8V	444 kbps	0.64 nJ/b core, IO
Trellis decoder	0.18 μm	.036mW @1.8V	4.44 Mbps	0.0082 nJ/b core, IO

Outlook for Analog Technology





References

[Win04] C. Winstead, J. Die, S. Yu, C. Myers, R. Harrison, and C. Schlegel, "CMOS Analog MAP decoder for an (8,4) Hamming code," *IEEE J. Solid State Cir.*, Vol. 29, No. 1, pp. 122–131, January 2004.

[Dai01] J. Dai, *Design Methodology for Analog VLSI Implementations of Error Control Decoders*, PhD thesis, University of Utah, 2001.

[Dai02] J. Dai, C.J. Winstead, C.J. Myers, R.R. Harrison, and C. Schlegel, "Cell library for automatic synthesis of analog error control decoders", *Proc. Int. Symp. Circuits and Systems*, vol. 4, pp. IV-481–IV-484, May 2002.

[Dem02] A. Demosthenous and J. Taylor, "A 100-Mb/s 2.8-V CMOS current-mode analog Viterbi decoder", *IEEE J. Solid-State Circuits*, Vol. 37, No. 7, pp. 904–910, July 2002.

[Gau03] V.C. Gaudet and P.G. Gulak, "A 13.3-Mb/s 0.35-μm CMOS analog turbo decoder IC with a configurable interleaver", *IEEE J. Solid-State Circuits*, Vol. 38, No. 11, pp. 2010–2015, November 2003.

[Hag98] J. Hagenauer and M. Winklhofer, "The analog decoder," *Proc. Int. Symp. Inform. Theory*, August 1998.

[Loe01] H.A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog VLSI", *IEEE Tran. Inform. Theory*, Vol. 47, No. 2, pp. 837–843, February 2001.

[Vor05] D. Vogrig, A. Gerosa, A. Neviani, A. Graell i Amat, G. Montorsi, and S. Benedetto, "A 0.35- m CMOS Analog Turbo Decoder for the 40-bit Rate 1/3 UMTS Channel Code", *IEEE J. Solid-State Circuits*, Vol. 40, No. 3, pp. 773–783, March 2005.

[Win03] C. Winstead, D. Nguyen, V.C. Gaudet, and C. Schlegel, "Low-voltage CMOS circuits for analog decoders", *Int. Symp. Turbo Codes*, pp. 271–274, Brest, France, September 2003.

[Win04] C. Winstead, *Analog Iterative Error Control Decoders*, PhD Thesis, University of Alberta, 2004.