Modern Coding in Practice

Keith M. Chugg

(kchugg@trellisware.com)
TrellisWare Technologies, Inc.
16516 Via Esprillo, Suite 300
San Diego, CA 92127

(chugg@usc.edu)
Communication Sciences Institute
Ming Hsieh EE Dept
Los Angeles, CA 90089-2565
Talk Summary

• Overview of Modern, Turbo-like Codes
  - Variations, characteristics
  - Good, TLC performance

• S-SCP Codes
  - Systematic RA codes
  - TrellisWare Codes

• Performance and Complexity
  - Hardware codec capabilities
  - Comparisons
Overview of Modern, Turbo-Like Codes
Theoretical Limits for QPSK/AWGN (BPSK/AWGN)

- **BW Efficiency (bits/sec)/Hz**
- **Eb/No(dB)**
- **AWGN Capacity**
- **QPSK/AWGN Capacity**
- **no coding**
- **(7,4) Hamming soft-in**
- **(7,4) Hamming hard-in**
- **Conv., 64-state (soft)**
- **Finite Block Size (BLER = 1e-4)**

- **k** = 65,536; 16,384; 4096; 2048; 1024; 512; 256; 128
A Wrong Turn in Coding?

• (Classic) Coding theorist's (practitioner's) lament
  - "All codes are good codes, except those we know about!"
• Is the (7,4) Hamming a bad code?
  - Not for it's size!
• To approach capacity, need ~ 2048 input bits
Classic vs. Modern Coding

- **Classic codes:**
  - “design bad (small) codes that we can decode optimally”

- **Modern codes:**
  - “design good codes that we can decode in a nearly optimal manner”

\[
P_b \leq \sum_{d \geq d_{\text{min}}} K_d Q \left( r d \frac{2E_b}{N_0} \right)
\]

Obviously, dominated by \( d_{\text{min}} \) term

\[\Rightarrow \] look for codes with large min. distance

Hmm... Can we make each \( K_d \) decay ~ 1/k or faster?

\[\Rightarrow \] find big codes with “thin” neighborhoods
Modern vs. Classic Code

Modern:
- fewer neighbors => smaller Kd

Classic:
- large d_min

- High SNR ==> d_min dominates
- Low SNR ==> Kd can be dominant
  - e.g., Kd < 1e-10 for all d...
Modern vs. Classic Coding

• We were designing for the wrong region...
Examples of Modern Codes

Parallel Concatenated Convolutional Codes (PCCCs) [Turbo Codes]

Serially Concatenated Convolutional Codes (SCCCs)

Hybrid Concatenated Convolutional Codes (HCCCs)

Low Density Parity Check Codes (LDPCCs)

Turbo Product Codes (TPCs)
Modern Code Analysis/Design

- Uniform Interleaver Analysis
  - What kind of codes work together?
  - How does error rate change with block size?

- SNR Threshold Optimization
  - Try to match specific constituent code properties to get convergence at lowest SNR
    - Irregular structures can improve thresholds
SNR Threshold Optimization


Uniform Interleaver Analysis


Uniform Interleaver Analysis

- Analyze union bound as block size tends toward infinity
  - Determine trends in BER, BLER
  - Determine design rules

\[
P_b \lesssim \sum_{d \geq d_{\text{min}}} K_d Q \left( \sqrt{\frac{r d E_b}{2 N_0}} \right)
\]

\[
K_d \sim C_d \left( \sum_{\alpha(d)} N^{\alpha(d)} \right)
\]

Asymptotic behavior:

\[
P_b \sim N^{\alpha_{\text{max}}}
\]

\[
P_{cw} \sim N^{\alpha_{\text{max}}+1}
\]

\[
\alpha_{\text{max}} = \max_d \alpha(d)
\]
Examples:

- PCCCs (w/ recursive encoders):
  - BER interleaver gain
  - No BLER interleaver gain
- SCCCs (w/ recursive inner code):
  - BER & BLER interleaver gain for do, min ≥ 3

\[ \alpha_{\text{max}} = -1 \]

\[ \alpha_{\text{max}} = -\left\lfloor \frac{d_{o,\text{min}} + 1}{2} \right\rfloor \]

Some constructions naturally have better floor properties - eg, SCCCs have lower floors than PCCCs.
Typical Floor/Threshold Trade-off

- BER
- Eb/No (dB)
- ~0.25 dB
- low threshold (e.g., PCCC)
- low-floor (e.g., SCCC)
- ~100-1000
Good TLC Performance

• Theoretical bounds
  - Capacity
    ‣ Infinite block size, zero error probability
  - Random coding bound
    ‣ Upper bound on BLER for ML decoding of random codes
  - Sphere packing bound
    ‣ Lower bound on BLER for ML decoding of any code
• Nontrivial to evaluate with a modulation constraint
• Use a pragmatic guideline
Finite Block-Size Performance - Pragmatic Guideline

- Symmetric Information Rate (SIR)
  - “Modulation constrained capacity”

- Use an SNR penalty for finite block size

\[
\left( \frac{E_s}{N_0} \right)_{\text{min, finite, dB}} = \left( \frac{E_s}{N_0} \right)_{\text{min, SIR, dB}} + \Delta_{dB}
\]

\[
\Delta_{dB} = \sqrt{\frac{20\eta (2\eta + 1) [10 \log_{10}(1/P_{cw})]}{k \ln(10) (2\eta - 1)}}
\]


8PSK Performance Limits

Theoretical Limits for 8PSK/AWGN

- AWGN Capacity
- 8PSK/AWGN Capacity
- TCM 64-state
- Finite Block Size (BLER = 1e-4)

BW Efficiency (bits/sec)/Hz vs. Eb/No(dB)

- k = 65,536; 16,384; 4,096; 2,048; 1,024; 512; 256; 128
16QAM Perf. Limits

Theoretical Limits for 16QAM/AWGN

BW Efficiency (bits/sec)/Hz

AWGN Capacity

16QAM/AWGN SIR

k = 65,536; 16,384; 4096; 2048; 1024; 512; 256; 128

Finite Block Size (BLER = 1e-4)
# Equivalent Modeling Conventions

<table>
<thead>
<tr>
<th>Model</th>
<th>Time (index)</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Diagram</td>
<td>implicit</td>
<td>implicit</td>
</tr>
<tr>
<td>Trellis</td>
<td>explicit</td>
<td>explicit</td>
</tr>
<tr>
<td>Graph</td>
<td>explicit</td>
<td>implicit</td>
</tr>
</tbody>
</table>

- **Block Diagram**: `implicit` values across all time indices.
- **Trellis**: `explicit` values at `k=0`, `k=1`, `k=2`, `k=1022`, `k=1023`.
- **Graph**: `explicit` values at `k=0`, `k=1`, `k=2`, `k=1022`, `k=1023`.

---

### FSM Diagrams

**Input**: `b[k]`  
**Output**: `c[k]`  

- **States** at `k=0`, `k=1`, `k=2`, `k=1022`, `k=1023`:
  - `b[k]` transitions are shown.
  - `c[k]` outputs are shown at each state.

---

**Footnote**: Mar. 2, 2009
(Iterative) Message Passing Decoding

- Given a model that has constraints and variables
  - Well-defined standard set of message-passing rules
  - Locally optimal processing
    ▸ Yields globally optimal results if graphical model is a tree
    ▸ Good heuristic if cycles are present -- “iterative message-passing algorithm” (iMPA)
  ✓ Modern, TLCs are decoding in this manner
Example Message Updates

- General rules
  - Compute configuration metric for all allowable configurations
  - Marginalize these over valid configurations with \( b=0, b=1 \)
    - Output this difference minus the incoming message

Example shown is for min-sum


S-SCP Codes
**S-SCP Structure & Examples**

(a) General S-SCP structure

\[ r = \frac{J}{J+Q} \]

- \( b_i \) to outer code
- \( d_j \) to Inner Parity Generator (IPG)
- \( kQ \) bits
- \( k \) systematic bits
- \( P = \frac{kQ}{J} \) parity bits
- \( p_m \)

(b) Systematic Repeat Accumulate

\[ r = \frac{J}{J+4} \]

- \( b_i \) to repetition code
- \( d_j \) to IPG
- \( P = \frac{kQ}{J} \) parity bits
- \( p_m \)

(c) TrellisWare Code

\[ r = \frac{J}{J+2} \]

- \( b_i \) to outer convolutional code
- \( d_j \) to IPG
- \( P = \frac{kQ}{J} \) parity bits
- \( p_m \)
TrellisWare Flexible S-SCP Examples

- “FlexiCode”
  - Outer Code: 4-state non-recursive (r=1/2)
  - Inner Code: 4-state recursive (r=1)
  - Very good floor properties

- Flexible LDPC (F-LDPC) Code
  - Outer Code: 2-state non-recursive (r=1/2)
    - 1+D, 1+D
  - Inner Code: 2-state recursive (r=1)
    - 1/(1+D) - same as Systematic RA
  - Extremely low complexity
TW Code Properties

• Good performance over all code rates and block sizes
  - r = J / (J + 2), flexible
  - finer rates by puncturing parity

• Uniform interleaver analysis:
  - Same as SCCCs
  - Unlike SCCCs, this can be achieved for very high codes rates
    ▸ Can achieve low error probabilities, even at small block sizes and high code rates

\[ \alpha_{\text{max}} = - \left\lfloor \frac{d_{o,\text{min}} + 1}{2} \right\rfloor \]
A New Class of Turbo-like Codes with Universally Good Performance and High-Speed Decoding

(MILCOM 2005)

Keith M. Chugg¹,², Phunsak Thienviboon², Georgios D. Dimou², Paul Gray², and Jordan Melzer¹

¹Communication Sciences Institute
Electrical Engineering Dept.
University of Southern California
Los Angeles, CA, 90089-2565
{chugg,jmelzer}@usc.edu

²TrellisWare Technologies, Inc.
16516 Via Esprillo, Ste. 300
San Diego, CA 92127–1708
{phunsak,gdimou,pgray}@trellisware.com


Iterative Decoding of TW Codes

channel metrics for systematic bits

channel metrics for parity bits

(a)

(b)
Systematic RA & F-LDPC Code Graphs

Sys-RA

(b) Outer Rep. Code

same performance, but Sys-RA uses 2x the interleaver size
F-LDPC Inner/Outer SISOS (same)

Accumulator (FBA) SISO

\[
\begin{align*}
F_{j+1} &= g(F_j, MI[a_j]) + MI[x_j] \\
B_j &= g(B_{j+1} + MI[x_j], MI[a_j]) \\
MO[a_j] &= g(F_j, B_{j+1} + MI[x_j]) \\
MO[x_j] &= F_{j+1} + B_{j+1} - MI[x_j]
\end{align*}
\]

Normalized add-compare-select operator

\[
\begin{align*}
g(x, y) &= \min(x, y) - \min(0, x + y) \\
&= \min(|x|, |y|)\text{sign}(x)\text{sign}(y) \\
&= \min(|x|, |y|)\text{sign}(x)\text{sign}(y) - \ln \left[ \frac{1 + \exp(-|x - y|)}{1 + \exp(-|x + y|)} \right] \\
&= \min^*(x, y) - \min^*(0, x + y) \\
&= \min(|x|, |y|)\text{sign}(x)\text{sign}(y) - \ln \left[ \frac{1 + \exp(-|x - y|)}{1 + \exp(-|x + y|)} \right]
\end{align*}
\]

- Essential same as LDPC Code check node processing
Systematic RA vs. TrellisWare F-LDPC

- Sys-RA with Q=4 has same performance as F-LDPC
  
  - Processing complexity
    - Sys-RA processes: $Qk = 4k$ trellis sections per iteration
    - F-LDPC processes: $k$ (outer) and $2k$ (inner) trellis sections per iteration
      ✔ F-LDPC has 25% less processing complexity
  
  - Memory complexity (interleaver only)
    - Sys-RA: $Qk = 4k$ storage elements
    - F-LDPC: $2k$ storage elements
      ✔ F-LDPC requires 50% less memory for messages exchanged
Sys-RA, F-LDPC as Structured-LDPCs

• Obtained by pulling all variable nodes to one side

Sys-RA:

\[
[S \mid H_b] \begin{pmatrix} p \\ b \end{pmatrix} = 0
\]

\[
S = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
\vdots & \ddots & 0 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix} (P \times P)
\]

F-LDPC:

\[
[S \ V \ 0] \begin{pmatrix} p \\ c \\ b \end{pmatrix} = 0
\]

\[
0 \ I \ G
\]
Systematic, Irregular RA - Structured LDPC Codes

- DVB-S2 and IEEE 802.11n, 802.16e codes both are based on Systematic, IR-RA codes
  - Allow for simple encoding
  - Decoding schedule options

![Diagram of LDPC code structure](image)
Systematic, Irregular RA - Structured LDPC Codes

\[ H = [ H_b \mid S ] \]

\[ \bar{Q} = \text{average column weight of } H_b \]

\[ J = \text{(average) row weight of } H_b \]

- Complexity varies linearly with Q-bar
  - irregular designs increase Q-bar
    - improved performance with software simulations (typical 0.1-0.3 dB)
    - often not realized in practice due to limited hardware resources
F-LDPC Performance & Complexity
Main Advantages of TW Codes (F-LDPC)

• Flexibility over rates $\geq 1/2$
• Flexibility over all block sizes
• Simple/high-speed implementation
• High code rates with low floors
• Universally good performance

- Hardware performance
  ‣ typically within 1 dB of the pragmatic guideline over all code rates and block sizes
  ✓ Recall best ‘academic’ point designs are 0.5 dB away
## F-LDPC as Compared to Sys-RA/LDPCs

<table>
<thead>
<tr>
<th>CODE</th>
<th>Comp. Complexity (# of accumulator trellis sections)</th>
<th>Memory Requirements (internal + channel)</th>
<th>Relative to F-LDPC (computation, memory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-LDPC</td>
<td>3k</td>
<td>2k + 4k</td>
<td>1, 1</td>
</tr>
<tr>
<td>Sys-RA (Q=4)</td>
<td>4k</td>
<td>4k+4k</td>
<td>1.3, 1.3</td>
</tr>
<tr>
<td>Sys, IR-RA (academic)</td>
<td>19.7 k</td>
<td>19.7k+4k</td>
<td>6.6, 4</td>
</tr>
<tr>
<td>LDPC (regular 3,6)</td>
<td>6k</td>
<td>6k + 4k</td>
<td>2, 1.6</td>
</tr>
<tr>
<td>Irregular LDPC (academic)</td>
<td>13k</td>
<td>13k+4k</td>
<td>4.3, 2.8</td>
</tr>
<tr>
<td>802.11n (Q-bar = 5.1)</td>
<td>5.1k</td>
<td>5.1k+4k</td>
<td>1.7, 1.5</td>
</tr>
<tr>
<td>DVB-S2 (Q-bar = 5)</td>
<td>5k</td>
<td>5k+4k</td>
<td>1.7, 1.5</td>
</tr>
</tbody>
</table>

*per iteration comp. complexity, k=input block size, r=1/2*

(memory requirements depend on architecture details - typically, more memory is required for decoder implementations not based on Sys.-RA structure)
Relation to Hardware Performance

• Number of Trellis sections processed per iteration translates to throughput
• Memory requirements affect size
• Characterize a codec by
  - decoded info bits/sec per logic-Hz
    ▸ bps/logic-Hz
  - performance at an achievable number of iterations
    ▸ best LDPC designs can perform slightly better in simulations, but gains are lost with hardware constraints
Why Add Complexity (irregularity)?

- Irregular designs have better asymptotic threshold
  - Typically small performance gains over F-LDPC at block sizes of interest (<= 0.2 dB)
  - Gains are often lost in practice due to
    - limited hardware resources
      - 15 iterations with F-LDPC vs. 9 iterations with DVB-S2
    - hardware approximations
      - Highly irregular designs more sensitive to min*-sum processing gains and hardware approximation errors

- Standards-based codes are selected based on academic-style simulations, not hardware testbeds
- TW has designed and implemented irregular F-LDPCs
Example

• Low-speed architecture
  - 1 trellis section per clock cycle
    ▸ F-LDPC: $1/(3i)$ bps/logic-Hz
    ▸ 802.11n, DVB-S2: $1/(5i)$ bps/logic-Hz
  - Assuming $i=10$ iterations, and 100 MHz clock
    ▸ F-LDPC: 3.3 Mbps
    ▸ 802.11n, DVB-S2: 2.0 Mbps

• May trade more logic for throughput...
  - Higher speeds require parallelism
  - Same proportional trade between codes
Example F-LDPC Rate Flexibility

- $k = 8000$ bits
Example F-LDPC Flexibility (k & r)
Example F-LDPC Flexibility (k & r)
F-LDPC Hardware Codec Performance - QPSK

![Graph showing performance curves for different coding rates](image)

- r = 2/3
- r = 4/5
- r = 8/9
F-LDPC Hardware Codec Performance - k=16K
Conclusions

- Modern (turbo-like) codes
  - Based on random-like code constructions
  - Iterative message-passing decoding (suboptimal)
- Many different constructions
  - Optimized for point designs
    - Gains often come with increased complexity
- Systematic with Serially Concatenated Parity (S-SCP)
  - Yield flexible, universally good TLCs
  - TrellisWare’s F-LDPC
    - Simplified version of Systematic Repeat Accumulate
IEEE (802.11n, 802.16) LDPC Structure

IEEE Cyclic Permutation Format, Z=3

<table>
<thead>
<tr>
<th></th>
<th>V1-3</th>
<th>V4-6</th>
<th>V7-9</th>
<th>V10-12</th>
<th>P1-3</th>
<th>P4-6</th>
<th>P7-9</th>
<th>P10-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1-3</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C4-6</td>
<td>-</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>C7-9</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C10-12</td>
<td>2</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

-’s represent the ZxZ all-zeros matrix
#’s represent permutations of the ZxZ identity matrix
(columns are cyclicly shifted # times to the right)

Standard Format of IEEE Matrix Above

- TrellisWare Technologies