STANFORD

Lecture 11

Outer Hard-Code Concatenation

February 15, 2024

JOHN M. CIOFFI

Hitachi Professor Emeritus (recalled) of Engineering

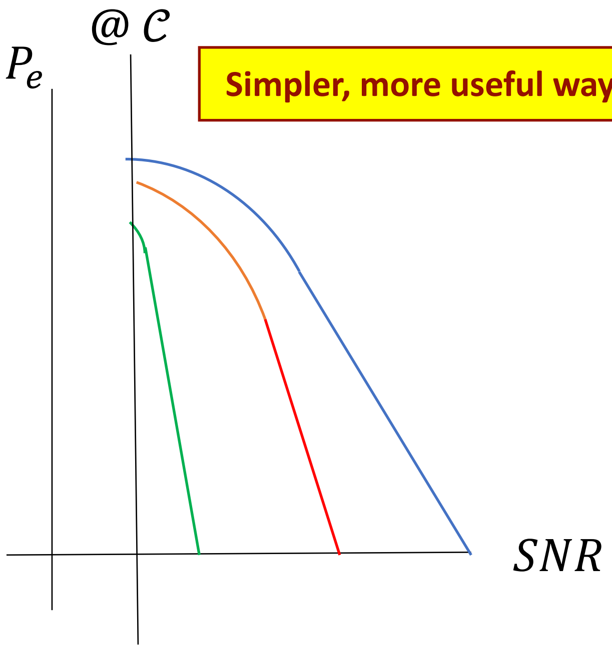
Instructor EE379A – Winter 2024

Announcements & Agenda

- Announcements
  - PS5 due 2/23

- PS4 Feedback
  - 9.5-20 hours
  - Some matlab complaints (diff in notation)
    - This is much harder to edit than students may realize.
    - Projects welcome in this area.
  - Short period (needed for test study)
    - PS8/final builds in longer time
  - Viterbi coverage was short
    - Torn here between Viterbi becoming obsolete, but older systems using it are deployed widely.
  - May delete it in future (constraint/iteration , GRAND)

- Today
  - Last 3 slides of L10 are for information only.
  - Deterministic Interleaving
  - Design with Reed Solomon to zero gap (nearly)
  - Cyclic Codes Overview
  - Retransmission - Error-Detecting Codes (CRC)



Simpler, more useful way to approach capacity

Deterministic Interleaving

Section 8.4

Redistribute the Inner Codes' errors

Outer encode → Interleave π → Inner encode → channel → Inner decode → π^{-1} De-interleave → Outer decode

- **Inner code** will make "whole-codeword" errors P_e. (This might be already a turbo or LDPC code.)
  - There are many **bit/subsymbol** errors correspondingly – i.e., an "error burst."
  - Error bursts also occur from nonstationary effects, such as:
    - random fades in wireless, or
    - impulse noise in wireline (or wireless).
- **Outer Code** design assumes that bursts are significantly separated (good inner code design, low $P_{e,inner} \sim 10^{-3\ to\ -7}$).
- **Deterministic interleaving** disperses these bursts evenly over **depth** $\mathcal{J}$ different codewords.
- Thus, $d_{free} \rightarrow \mathcal{J} \cdot d_{free}$, and really the entire distance d_i distribution increases by $\mathcal{J}$.
  - This *interleave gain* applies to a burst, not overall; but does thereby add ~ 0.5 - 1 dB more coding gain.
  - The aggregate design operates close to capacity and $P_e \rightarrow 0$
  - $\frac{dP_e}{dSNR} \rightarrow -\infty$; Pe versus energy becomes very steep/sensitive.
    - So operation at/very-near capacity is requires highly stationary channel to be effective.
    - Whence our EE379 "margin" concept. (Design for capacity at presumed larger noise, but operate with the actual noise.)

Formal (deterministic-interleaver) Depth

- depth **Definition 8.6.1** **[Interleaver Depth]** *The* **depth** $\mathcal{J}$ *of an interleaver is the minimum separation in subsymbol periods at the interleaver output between any two subsymbols that are adjacent at the interleaver input.*

$$\mathcal{J} = \min_{k=0,\ldots,L-1} \left| \pi^{-1}(k) - \pi^{-1}(k+1) \right|$$

- period **Definition 8.6.2** **[Interleaver Period]** *The* **period** L *of an interleaver is the shortest time interval for which the re-ordering algorithm used by the interleaver repeats.*

ss ≜ subsymbol

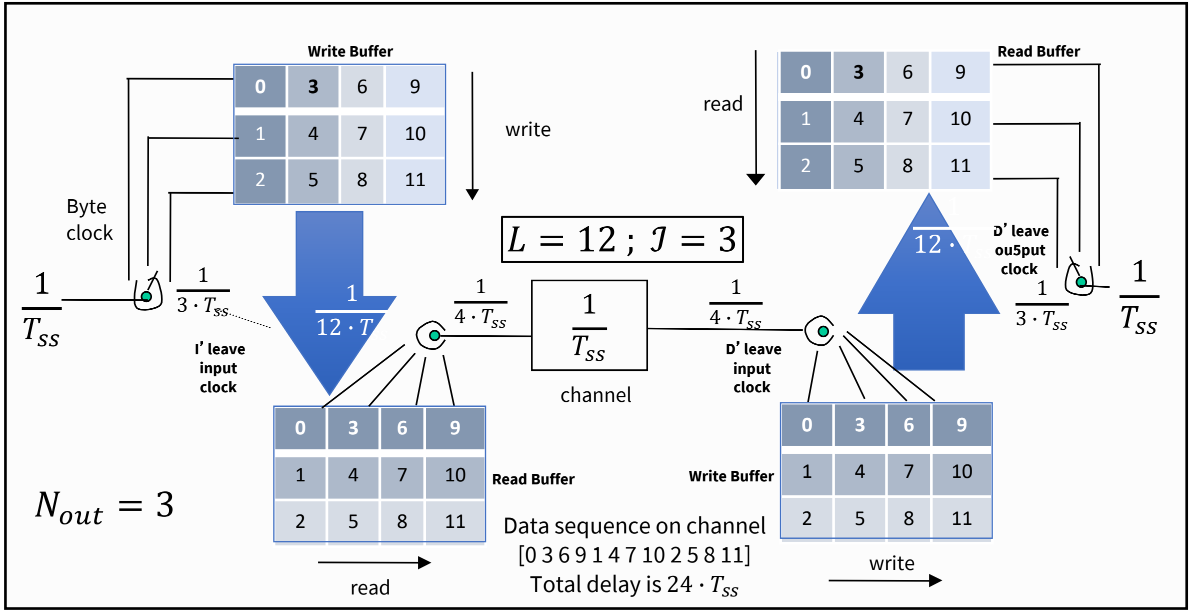
- Distance magnification is $d_{free} \to \mathcal{J} \cdot d_{free}$; but introduces delay $\propto \mathcal{J} \cdot L$.
- The outer code is typically cyclic, specifically Reed Solomon (coming) and not binary (usually ss = bytes).
- System-design perspective:
  - Pick an RS code with high rate $r \to R = K/N$ and just enough distance (so rate is high) to meet target $[P_e \quad \bar{P}_b]$.
    - Design outer code for inner-code's eventual hard-decoded output, and model as a symmetric DMC.
  - Design for "not too much delay in the interleaving and de-interleaving."

Classical Block Interleaver

- Two transmit memories: read and write

$$G = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}$$

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



- Write Buffer inputs 4 blocks of 3 subsymbols each.
- Read Buffer outputs 3 blocks of 4 subsymbols each.
- De-interleave reverses interleaver.
- Delay is 12 units on each side, so 24 total.

At least $\mathcal{J} = 3$ subsymbols between adjacent de-interleaver outputs, e.g. 11 and 10 are 4 apart. (delay ss 11 by 12 ss times to avoid being next to next period's ss 0).

We could reverse to $\mathcal{J} = 4$ with $N_{out} = 4$.

Minimum (block-ileave)Memory Implementation

First 6		2nd 6	
0	3	6	9
1	4	7	10
2	5	8	11

write order 0,1,2,3,4,5

write order 6,7,8,9,10,11

$$G = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix}$$

m	read	write	cell
0	---	----	pass
1	Past 3	Currrent 1	A
2	Past 6	Current 2	B
3	Past 9	Current 3	C
4	Current 1	Current 4	A
5	Current 4	Current 5	A
6	Past 7	Current 6	D
7	Past 10	Current 7	E
8	Current 3	Current 8	C
9	Current 5	Current 9	A
10	Current 8	Current 10	C
11	Past 11	Current 11	F

- Overwrite memory cells as they become available,
- See right-side table.

6 Memory Cells Needed

- This is ½ delay (12 end-to-end, not 24) and ¼ memory of classical block interleaver (12 instead of 48)

Convolutional Interleaver Generator

$$x_k \longrightarrow \boxed{\begin{array}{c}\text{Interleaver}\\ \pi(k)\\ \text{period } L\\ G(D)\end{array}} \longrightarrow \tilde{x}_k$$

$r = 1$

$$\boldsymbol{X}(D) = \sum_{m=0} [\boldsymbol{x}_{mL+L-1} \quad \cdots \quad \boldsymbol{x}_{mL+1} \quad \boldsymbol{x}_{mL}] \cdot D^m$$

$$\tilde{\boldsymbol{X}}(D) = \sum_{m=0} [\tilde{\boldsymbol{x}}_{mL+L-1} \quad \cdots \quad \tilde{\boldsymbol{x}}_{mL+1} \quad \tilde{\boldsymbol{x}}_{mL}] \cdot D^m = \boldsymbol{X}(D) \cdot G(D)$$

- $G(D)$ must be causal linear; D corresponds to a delay of one interleaver period.
  - If $G(D) = G(0)$, then block interleaver, otherwise a convolutional interleaver.
  - Subsymbols interleaved may themselves be vectors.

- A period has L subsymbols within it. D delays one period, D_{ss} delays one subsymbol period.

- To relate roughly to an ss-based convolutional code, $D \rightarrow D_{ss}^L$, a period is L subsymbol periods.

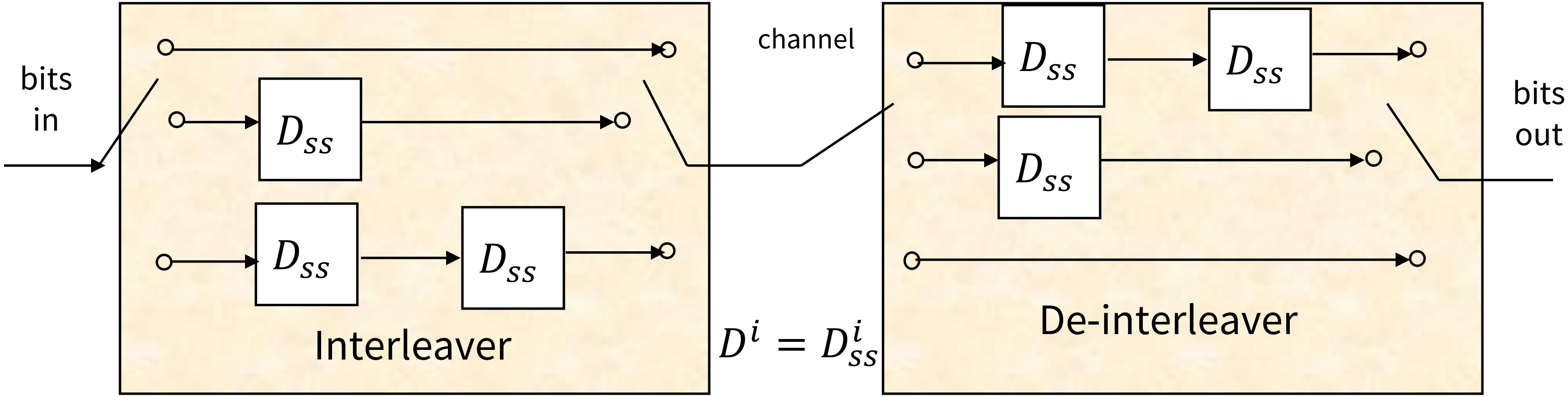
$$\boldsymbol{X}(D_{ss}) = \left[D_{ss}^{L-1} \cdot x_{L-1}(D)\,\Big|_{D=D_{ss}^L} \quad D_{ss}^{L-2} \cdot x_{L-2}(D)\,\Big|_{D=D_{ss}^L} \quad \ldots\; x_0(D)\,\Big|_{D=D_{ss}^L}\right]$$

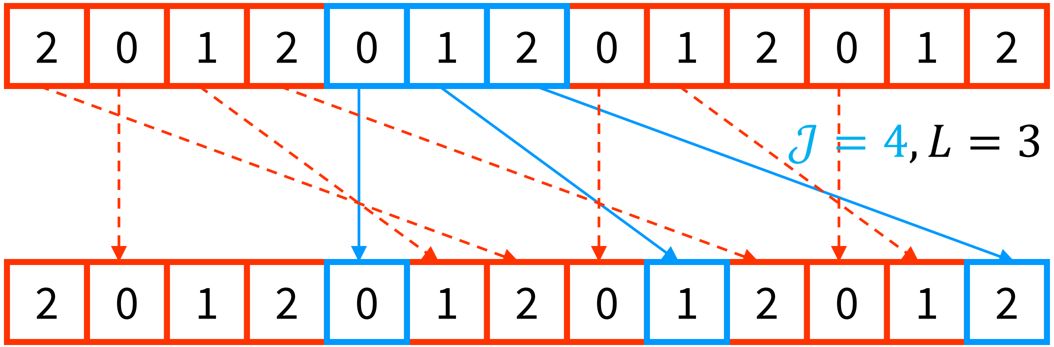
Specific examples in following slides

Convolutional/Triangular Interleaver, $\mathcal{J} = 4, L = 3$

- ~ delay/2 and memory/2 w.r.t. block

$\Delta_i = i \cdot L = i \cdot (\mathcal{J} - 1)$ symbol periods $i = 0, \ldots, L-1$



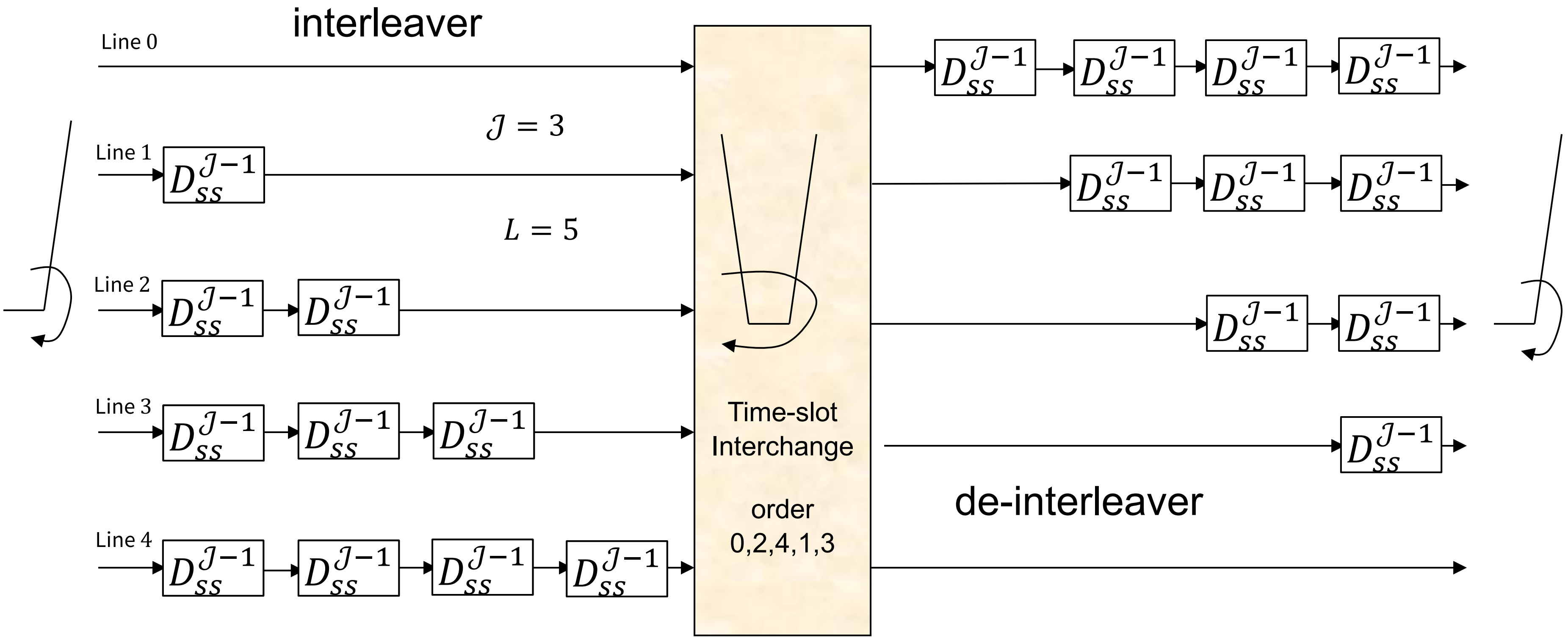


$$G(D_{ss}) = \begin{bmatrix} D_{ss}^{L-1} & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & D_{ss} & \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$6 = delay = \underbrace{L}_{\mathcal{J}-1} \cdot (L-1) \cong \underset{12}{block/2}$$

$$6 = memory = L^2 - L \cong \underset{24}{block/2}$$

Convolutional Interleavers, coprime L , $\mathcal{J}$



$$delay = (\mathcal{J}-1)\cdot(L-1) = 8 \text{ subsymbol periods}$$

- The delays are in D_{ss} , not D . It still looks triangular, except for the time-slot interchange order.
  - Is not triangular with D , see also example with $\mathcal{J} = 4$; $L = 5$ in Section 8.6.1.3.

Minimum Memory Requirement in cells

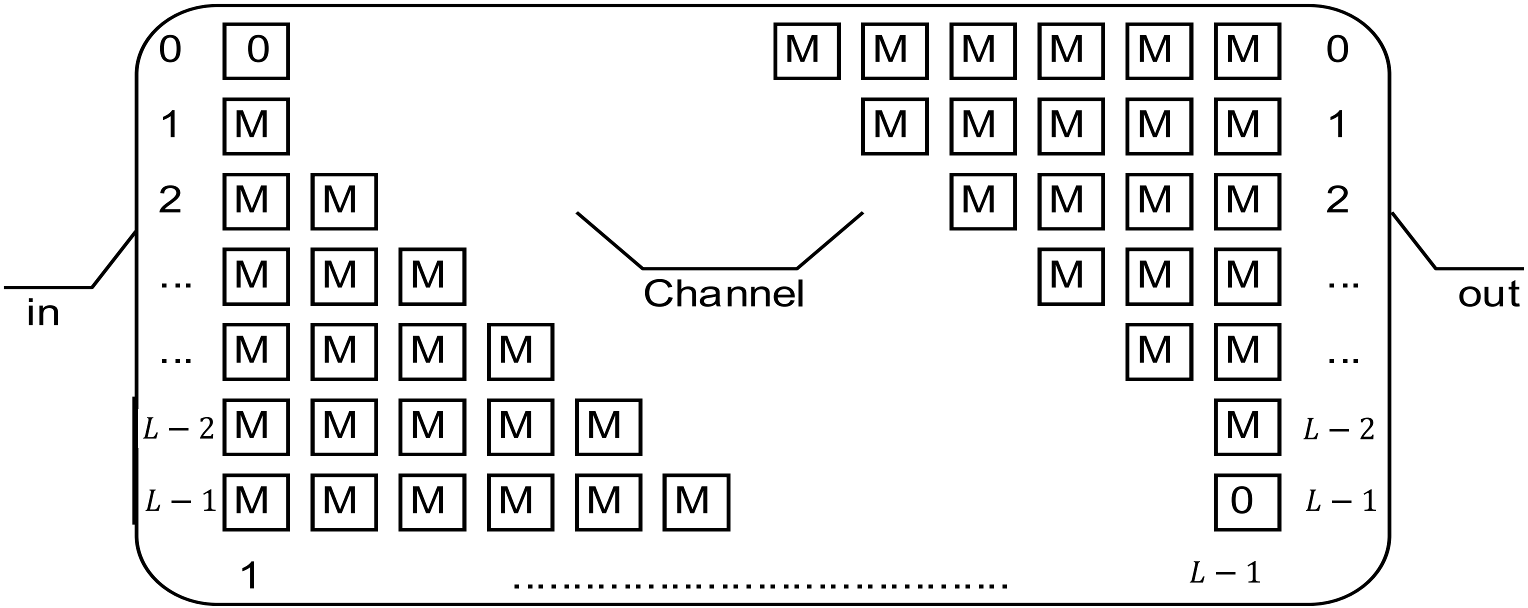
Table 2 for J=3 and L=5															
L/t	0	1	2	3	4	0'	1'	2'	3'	4'	0''	1''	2''	3''	4''
0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1	-	B1	B1	-	-	-	B1'	B1'	-	-	-	B1''	B1''	-	-
2	-	-	B2	B2	B2	B2	-	B2'	B2'	B2'	B2'	-	B2''	B2''	B2''
3	-	-	-	B3	B3	B3	B3	B3	B3 B3'	B3'	B3'	B3'	B3'	B3'' B3'	B3''
4	-	-	-	-	B4	B4	B4	B4	B4	B4 B4'	B4 B4'	B4 B4'	B4'	B4'	B4'' B4'
CELL1	-	B1	B1	B3	B3	B3	B3	B3	B3	B4'	B4'	B4'	B4'	B4'	B4'
CELL2	-	-	B2	B2	B2	B2	B1'	B1'	B3'	B3'	B3'	B3'	B3'	B3'	B4''
CELL3	-	-	-	-	B4	B4	B4	B4	B4	B4	B4	B4	B2''	B2''	B2''
CELL4								B2'	B2'	B2'	B2'	B1''	B1''	B3''	B3''

- Can do it with $\frac{1}{2} \cdot (\mathcal{J} - 1) \cdot (L - 1)$ CELLS in general (so yet another factor of 2 less)

Generalized Triangular

- Group M subsymbols

$$\boxed{\text{M}} = \boxed{D_{ss}^{M\cdot L}}$$



ITU Generalized Triangular

- Used some wireline standards

Parameter	Value
Interleaver block length (K)	$K = L$ *subsymbols (equal to or divisor of* N*)*
Interleaving Depth ($\mathcal{J}$)	$\mathcal{J} = M \cdot K + 1$
(De)interleaver memory size	$M \cdot K \cdot K \cdot ({}^{K-1}/_{2})$ subsymbols
Correction capability (block code that corrects t symbol errors) With *q=N/K*)	$\left\lfloor \frac{t}{q} \right\rfloor \cdot (M \cdot K + 1)$ *subsymbols* $\left\lfloor \frac{t}{q} \right\rfloor \cdot (\mathcal{J})$
End-to-end delay	$M \cdot K \cdot (K-1)$ *subsymbols*

Rate (Mbps)	Interleaver parameters	Interleaver depth (*J*)	(De)interleaver memory size	Erasure correction	End-to-end delay
50x1024	$K = 72$ $M = 13$	937 blocks of 72 bytes	33228 bytes	3748 bytes 520 ns	9.23 μs
24x1024	$K = 36$ $M = 24$	865 blocks of 36 bytes	15120 bytes	1730 bytes 500 ns	8.75 μs
12x1024	$K = 36$ $M = 12$	433 blocks of 36 bytes	7560 bytes	866 bytes 501 ns	8.75 μs
6x1024	$K = 18$ $M = 24$	433 blocks of 18 bytes	3672 bytes	433 bytes 501 ns	8.5 μs
4x1024	$K = 18$ $M = 16$	289 blocks of 18 bytes	2448 bytes	289 bytes 501 ns	8.5 μs
2x1024	$K = 18$ $M = 8$	145 blocks of 18 bytes	1224 bytes	145 bytes 503 ns	8.5 μs

Design with Reed Solomon Codes

Section 8.6.2

Channel is typically the **SDMC**, Symmetric Discrete Memoryless Channel

Block (Outer) Code Performance

- The **codeword error probability** is

$$P_e = \sum_{i=\left\lfloor \frac{d_{free}+1}{2} \right\rfloor}^{N} \binom{N}{i} \cdot p_{ss}^i \cdot (1-p_{ss})^{N-i} \quad .$$

  - p_{ss} is the **subsymbol (byte) error** rate on the "hard" SDMC $\approx \tilde{b} \cdot \bar{P}_b$; hard subsymbol decisions.

- Outer code's $\bar{P}_b$:

$$\bar{P}_b = \frac{2^{\tilde{b}-1}}{\left(2^{\tilde{b}}-1\right) \cdot N} \sum_{i=\left\lfloor \frac{d_{free}+1}{2} \right\rfloor}^{N} i \cdot \binom{N}{i} \cdot p_{ss}^i \cdot (1-p_{ss})^{N-i}$$

$$\frac{\textit{half } C \textit{ points have a bit incorrect, on average}}{\textit{Total number of points } - \textit{correct point}}$$

- Semi-soft direct Gray-Map to 2^m-ary subsymbol (SQ-QAM/PAM, … without BICM) reduces to ($\tilde{b} > 2$):

$$\bar{P}_b = \frac{1}{\tilde{b} \cdot N} \sum_{i=\left\lfloor \frac{d_{free}+1}{2} \right\rfloor}^{N} i \cdot \binom{N}{i} \cdot p_{ss}^i \cdot (1-p_{ss})^{N-i}$$

$$\frac{\textit{Gray has only 1 bit on each symbol error}}{\textit{Total number of bits in } C}$$

Reed Solomon Code Performance

- Typically, RS codes are ss=byte oriented or $GF(256)$ with max codeword length $N_{out} \leq 255 = 2^8 - 1$ bytes.
- $\tilde{b} = m$ in $GF(2^m)$ more generally ($\tilde{b}$ = 8 for bytes).

- There are P parity bytes (preferred implementation is systematic).
  - So $K = N_{out} - P$ information bytes,
  - $r = R = \frac{K}{N_{out}}$, &
  - $d_{free} = P + 1$, so if $P \in 2\mathbb{Z}^+$ (even), then RS ML decoder corrects $\left\lfloor \frac{d_{free}-1}{2} \right\rfloor = {}^{P}/_{2}$ erred subsymbols.

- To correct error bursts, use interleave depth $\mathcal{J}$, so that effectively $d_{free} \to d_{free} \cdot \mathcal{J}$, or correct ${}^{P \cdot \mathcal{J}}/_{2}$,
  - as long as error bursts are sufficiently separated.

- If burst-length = inner codeword length N_{in}, then select $\frac{P}{2} \cdot \mathcal{J} \geq \frac{N_{in}}{2}$ roughly, so $\mathcal{J} \geq \frac{N_{in}}{P}$.
  - So, design selects: N_{out}, P, and $\mathcal{J}$.
  - But larger depth means more memory and more delay – and also, bursts must be sufficiently separated!
- Higher P corrects more errors, but reduces the rate $r = R = \frac{N_{out}-P}{N_{out}}$.

- Usually pick maximum (or close to it) $N_{out} \leq 2^m - 1$ (255 for bytes).
  - Clearly $N_{out} = 2^m - 1$ yields highest rate for any given P.
  - But, there are also more chances for errors to occur with larger N_{out}, and d_{free} remains same even if $N_{out} < 2^m - 1$.

Example

- **Inner code** is LDPC with $N_{in} = 1000$ bytes (so $n = 8000$ bits or 1kB).
- **Delay specification:** The bit rate is R =8 Gbps (1 GB/s); an inner codeword occurs every 1 μs.
  - The specification's maximum delay is 1 ms, so 1000 outer codewords (1MB) in 1 ms.
  - Then 1MB $\cong \underbrace{250}_{N_{out}} \cdot \mathcal{J}\ bytes$. Thus, $\mathcal{J}$<4000 maintains sub-ms delay.
- **Error-correction:** Inner system has $P_e = 10^{-3}$.
  - Inner decoder error bursts of up to 1000 erred bytes each arrive every 1ms, on average.
  - To correct the error burst of 1000 bytes using $\frac{P}{2} \cdot$ erred bytes per codeword means:
  - the RS code needs $P = 20$ parity bytes and $\mathcal{J}$ = 100.
  - $r = R = \frac{230}{250}$, so a fairly high rate will cause almost no errors with depth 100 (and delay 25 μs).
- This design should cause high reliability (larger coding gain in effect or really very low $\bar{P}_b$) if
  - The inner system satisfies $P_e = 10^{-3}$.
- Expect rapid degradation if inner system has slight increase in error probability (slight noise increase)
  - This is true of any system with $\Gamma \rightarrow 0$ dB (which is often why positive noise margin is also a design objective).

Matlab RS Encoder Program

- The inputs are m-bit elements in $GF(2^m)$
- This means they must be specially set in matlab to be elements in such a field using the gf command.
- As an example with $m = 3$ so GF(8)
  - There are $K = 4$ input bytes / codeword
  - $N = 7$ output bytes include the $P = 3$ parity bytes.

```
rsenc Reed-Solomon encoder.

  CODE = rsenc(MSG,N,K) encodes the message in MSG using an (N,K) Reed-
  Solomon encoder with the narrow-sense generator polynomial. MSG is a
  Galois array of symbols over GF(2^m). Each K-element row of MSG
  represents a message word, where the leftmost symbol is the most
  significant symbol. If N is smaller than 2^m-1, then rsenc uses a
  shortened Reed-Solomon code. Parity symbols are at the end of each word
  in the output Galois array code.
  --- deleted long comment on polynomal specification, allows more than
       Matlab's default RS polynomial to be used ----

  CODE = rsenc(...,PARITYPOS) specifies whether rsenc appends or prepends
  the parity symbols to the input message to form code. The string
  PARITYPOS can be either 'end' or 'beginning'. The default is 'end'.
```

```
Examples:
    N=7; K=3;                   % Codeword and message word lengths
    m=3;                        % Number of bits per symbol
    msg  = gf([5 2 3; 0 1 7],m);     % Two K-subsymbol message words
    code = rsenc(msg,N,K);      % Two N-subsymbol codewords
```

```
genpoly = rsgenpoly(N,K);       % Default generator polynomial
 code1 = rsenc(msg,N,K,genpoly);  % code and code1 are the same codewords
genpoly2 = rsgenpoly(N,K,primpoly); % primitive poly is octal G(D), see L11:21-25
```

```
>> msg
 = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
Array elements =
   5   2   3
   0   1   7
>> code
= GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
Array elements =
   5   2   3   5   4   4   2
   0   1   7   6   6   0   7
```

Matlab RS Decoder Program

- It accepts N bytes of (de-interleaved) channel output and decodes them.
- Result is correct if $\leq P/2$ erred bytes.
- The decoder algorithm is basically a pseudoinverse in a finite field:
  - It's nontrival.
  - See text or EE387.
  - It is Max Likelihood for SDMC.

```
N=7; K=3;                    % Codeword and message word lengths
    m=3;                     % Number of bits per symbol
    msg = gf([7 4 3;6 2 2;3 0 5],m)   % Three k-symbol message words
msg = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
  7  4  3
  6  2  2
  3  0  5
code = rsenc(msg,N,K);
  7  4  3  7  0  0  4
  6  2  2  7  6  7  3
  3  0  5  5  6  0  6
```

rsdec Reed-Solomon decoder.

DECODED = rsdec(CODE,N,K) attempts to decode the received signal in CODE using an (N,K) Reed-Solomon decoder with the narrow-sense generator polynomial. CODE is a Galois array of symbols over GF(2^m), where m is the number of bits per symbol. Each N-element row of CODE represents a corrupted systematic codeword, where the parity symbols are at the end and the leftmost symbol is the most significant symbol. If N is smaller than 2^m-1, then rsdec assumes that CODE is a corrupted version of a shortened code.

```
% Add 1 error in the 1st word, 2 errors in the 2nd, 3 errors in the 3rd
>> errors = gf([3 0 0 0 0 0 0;4 5 0 0 0 0 0;6 7 7 0 0 0 0],m);
>> codeNoi = code + errors
  = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
  4  4  3  7  0  0  4
  2  7  2  7  6  7  3
  5  7  2  5  6  0  6
```

```
[dec,cnumerr] = rsdec(codeNoi,N,K) % Decoding failure : cnumerr(3) is -1
dec = GF(2^3) array. Primitive polynomial = D^3+D+1 (11 decimal)
  7  4  3
  6  2  2
  5  7  2
cnumerr =   % recall dfree=5 for this code
   1    % corrected one error
   2    % corrected two errors
  -1    % detects error
```

Cyclic Code Basics

Section 8.4 and Appendix B

See also Chap 8 References [30] Blahut book and [31] Gill's EE387 Class Notes

Galois Field with $p = 2^m$

- $GF(2^m) = \{0,1,\dots,2^m-1\}$ - but elements are viewed as binary polynomials of degree m.
  - Addition/multiplication is modulo a degree-m prime **binary** polynomial.
  - $g(D) = g_0 + g_1 \cdot D + \cdots + g_{m-1} \cdot D^{m-1}$ has no factor in $GF(2\,)$ but itself is factor of $D^{2^m-1} + 1 = 0$, a root of 1.
    - This D is for a binary polynomial.

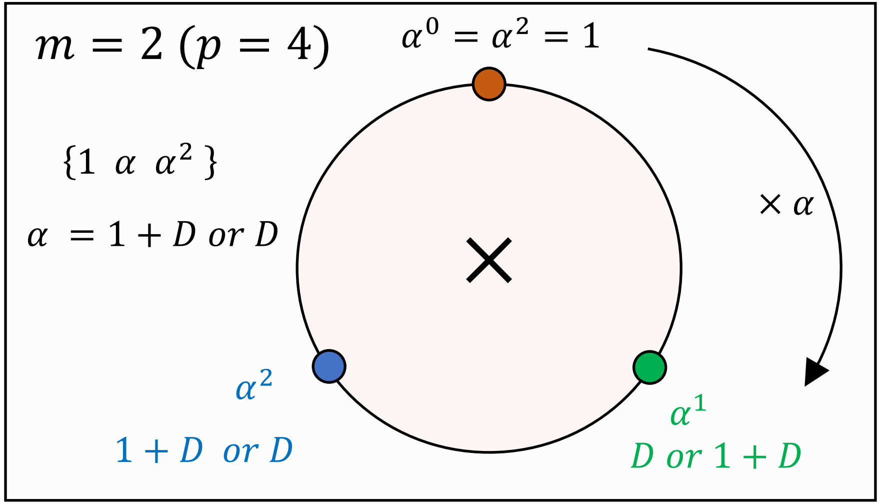
$$GF(2^m) = \{0 \;\; 1 \;\; \alpha \;\; \alpha^2 \;\; \dots \;\; \alpha^{2^m-2}\,\}$$

- Multiplication is modulo this prime polynomial.

- So multiply and set $g(D) = 0$

$$x(D) \cdot y(D) = d(D) \cdot g(D) + r(D)$$

$$\big(x(D) \cdot y(D)\big)_{g(D)} = r(D)$$



See example multiplication tables in Appendix B.1, as well as back-up slides

Cyclic Codes

- Every codeword is cyclic shift of others.
  - Subsymbols are elements in $GF(2^m)$.
  - More generally, $GF(p^m)$, see EE387.
- If $v(D) \in C$, then $\left(D^i \cdot v(D)\right)_{1-D^N} \in C$
  - Right circular shift by i places.

$$v(D) = v_0 + v_1 \cdot D + \cdots + v_{N-1} \cdot D^{N-1}$$

$$v_n \in GF(2^m)\,; \;\; n = 0, \ldots, N-1 \text{, so } v(D) \in [GF(2^m)]^N$$

$$\left(D \cdot v(D)\right)_{1-D^N} = v_{N-1} + v_0 \cdot D + \cdots + v_{N-2} \cdot D^{N-1}$$

- Some (like Reed Solomon) have $d_{free} = N - K + 1$; MDS code (meets Singleton Bound).
- Further, any $GF(2^m)$ linear combination of codewords (mod $1 - D^N$) is $\in C$.
  - $GF(2^m)$ defines the arithmetic, while an irreducible polynomial $G_j(D)$ defines the code ….
- $1 - D^N = \prod_{j=1}^{J} G_j(D)$ where each $G_j(D)$ is irreducible polynomial in $GF(2^m)$.
  - Clearly

$$\left(G_j(D) \cdot H_j(D)\right)_{1-D^N} = 0 \text{ where } H_j(D) = \prod_{i \neq j} G_i(D).$$

Note: The irreducible polynomial is NOT the Same binary polynomial used to define arithmetic in $GF(2^m)$ that was vector of bits This $G(D)$ is for a vector of bytes/subsymbols

- A cyclic code can be defined by each $G_j(D)$, with degree determining $N - K$, as
  - $C_i(D) = D^{N-K} \cdot u(D) + \left(D^{N-K} \cdot u(D)\right)_{G_j(D)}$ - delay the input ss's by $N-K$ and add the remainder in remaining $N-K$ positions.

Cyclic Code Continued

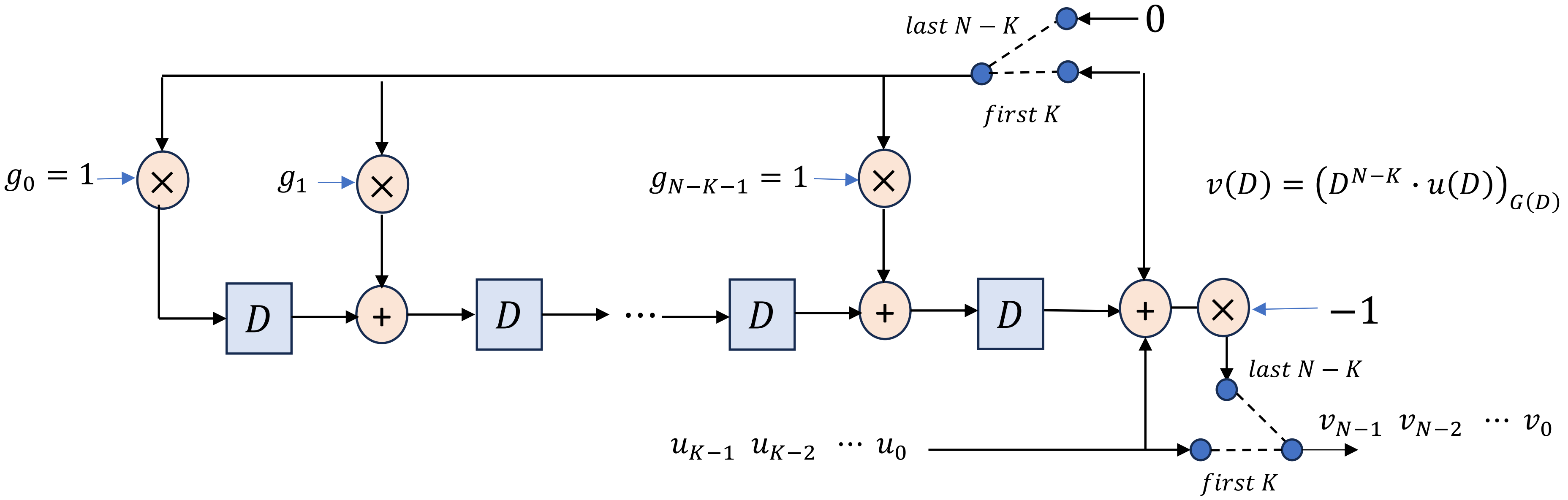
- $C_i(D)$ is cyclic because $v_{N-1} + v_0 \cdot D + \cdots + v_{N-2} \cdot D^{N-1} = D \cdot v(D) + v_{N-1} \cdot (1 - D^N)$

- Since $G\,(D)$ divdes both $v(D)$ and $(1 - D^N)$, then $\left(D^j \cdot v(D)\right)_{1-D^N}$ is also a codeword (any j).

- $H\,(D) = \frac{1-D^N}{G(D)}$ is parity polynomial
  - corresponding to an $(N, N-K)$ dual cyclic code with generator $D^K \cdot H(D^{-1})$
    - So unlike convolutional code where $H\,(D)$ is both parity matrix and dual code, with cyclic-generator simplifications for cyclic block codes, the dual code essentially reverses time w.r.t. $H\,(D)$.
    - This time reversal corresponds to circular convolution in $[GF\,(2^m)]^N$
- Syndrome calculation is then $\left(y(D) \cdot D^K \cdot H(D^{-1})\right)_{G(D)} = s(D) = \left(e(D) \cdot D^K \cdot H(D^{-1})\right)_{G(D)}$
  - ML decoder finds minimum Hamming weight $e(D)$ as solution (often nontrivial to find).

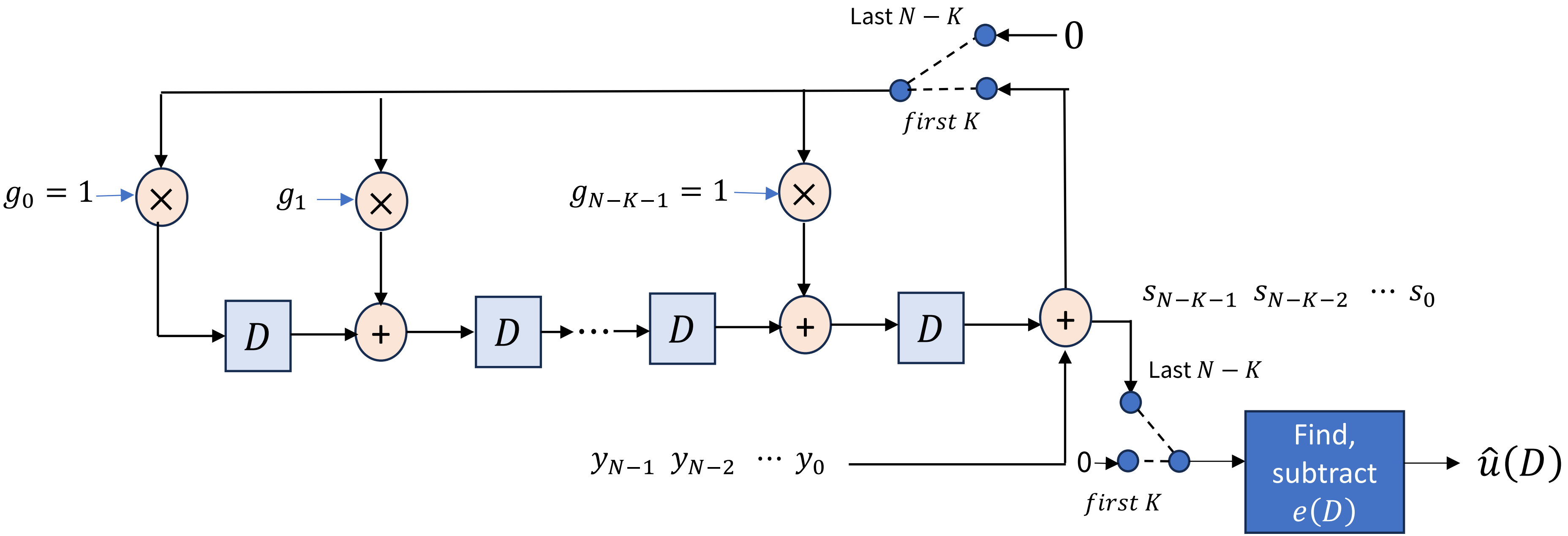
Encoder Circuit

- $G(D)$ is the cyclic code's generator (like convolutional) prime polynomial:



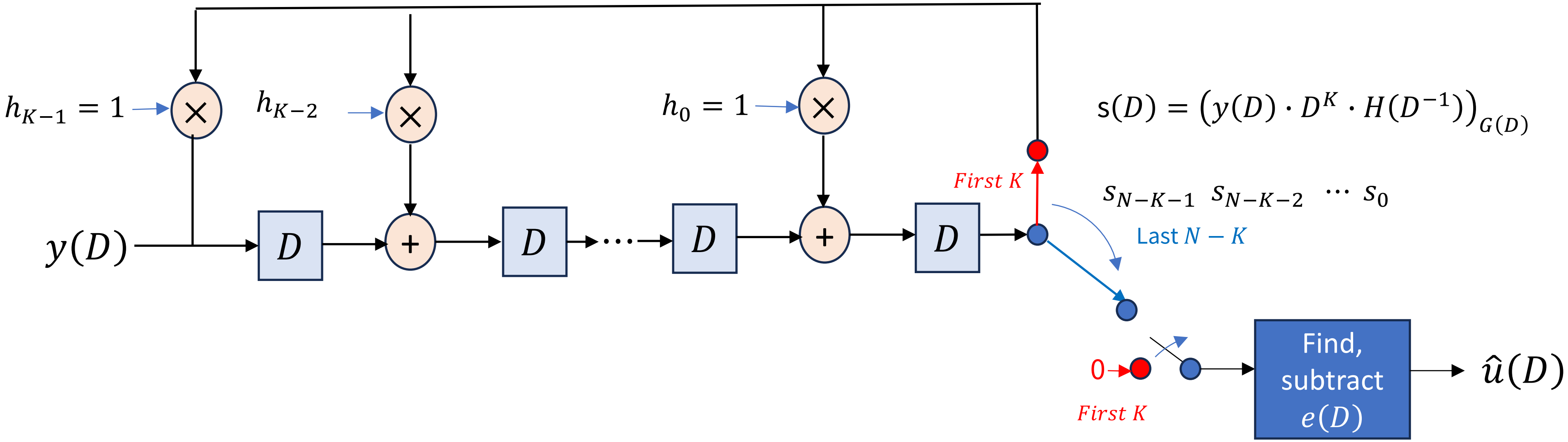
- $G(D)$ is the cyclic code's generator (like convolutional) prime polynomial with degree $N-K$.
- $D^{N-K} \cdot u(D) = q(D) \cdot G(D) + R(D)$ where $R(D)$ contains parity bytes/subsymbols.
- By subtracting $R(D)$, this encoder's output becomes a multiple of $G(D)$.

Decoder Circuit using G(D)



- $s(D)$ is the syndrome and equivalent to $v \cdot H$, which is zero if no errors w.r.t. any codeword.
- $s(D) = \left(e(D) \cdot D^K \cdot H(D^{-1})\right)_{G(D)}$, so the ML decoder must find smallest (w_H) $e(D)$ that causes $s(D)$.
- Then $\hat{u}(D) = D^{K-N} \cdot \left(y(D) - e(D)\right)$ - the decoder ignores any negative-power $D^{i<0}$ terms.
- Dark Blue Box is nontrivial for cyclic codes (Berlekamp-Massey, Forney, ….) – finite-field pseudoinverse,
  - which has structure that avoids a huge list-based ML decoder's complexity, unlike a more general block code might need.

Decoder Circuit using H(D)



- $G(D)$ or H(D) other will be simpler for any specific code.

Reed Solomon Generators (Cyclic Code)

- The blocklength is $N = 2^m - 1$;
  - but can reduce K and N together by same number of ss, keep P constant.

- $2t = N - K$ or $d_{free} = N - K + 1$ (achieve Singleton Bound Maximum)
  - $t =$ number of errors corrected.

- For any primitive element $\alpha \in GF(2^m)$:

$$G(D) = \prod_{i=1}^{N-K} (D + \alpha^i)$$

- Error prob for SDMC with subsymbol hard error P_{ss}

$$P_e \leq \sum_{i=t+1}^{N} \binom{N}{i} \cdot P_{ss}^i \cdot (1 - P_{xs})^{N-i}$$

$$P_{e,ss} \leq \sum_{i=t+1}^{N} \frac{i}{N} \cdot \binom{N}{i} \cdot P_{ss}^i \cdot (1 - P_{xs})^{N-i}$$

$$\bar{P}_b = \frac{2^{m-1}}{2^m - 1} \cdot P_{e,ss}$$

$$N_{e,i} = \binom{N}{i} \cdot N \cdot \sum_{j=0}^{i-d_{free}} (-1)^j \cdot \binom{i-1}{j} \cdot (N+1)^{i-j-d_{free}}$$

Retransmission – Error-Detecting Codes

Section 8.6.3

CRC Error Detection and Retransmission

- Cyclic Redundancy Check codes are (usually) binary and only detect errors (so $s(D) \neq 0$).
  - CRCs mostly use simple binary versions of the previous encoders/decoders.
  - Table below lists some with $d_{free} = 4$ and $n_{max} = 2^{2^r} - 1$.
- These detect:
  - all single and 2-bit errors, and also any odd number of bit errors. The $(D+1)$ factor forces even distance between codewords.
  - any burst of length $\leq n-k$ (because this is the length of $g(D)$ - such a burst is not divisible by $g(D)$).

Name	$g(D)$	factored
CRC-7	$D^7 + D^6 + D^4 + 1$	$(D^4 + D^3 + 1) \cdot (D^2 + D + 1) \cdot (D+1)$
CRC-8	$D^8 + D^2 + D + 1$	$(D^7 + D^6 + D^5 + D^4 + D^3 + D^2 + 1) \cdot (D+1)$
CRC-12	$D^{12} + D^{11} + D^3 + D^2 + D + 1$	$(D^{11} + D^2 + 1) \cdot (D+1)$
CRC-16 USA	$D^{16} + D^{15} + D^2 + 1$	$(D^{15} + D + 1) \cdot (D+1)$
CRC-16 Euro	$D^{16} + D^{15} + D^5 + 1$	$(D^{15} + D^{14} + D^{13} + D^{12} + D^4 + D^3 + D^2 + D + 1) \cdot (D+1)$
CRC-24	$D^{24} + D^{23} + D^{14} + D^{12} + D^8 + 1$	$(D^{10} + D^8 + D^7 + D^6 + D^5 + D^4 + D^3 + D + 1) \cdot$ $(D^{10} + D^9 + D^6 + D^4 + 1) \cdot (D+1)$
CRC-32	$D^{32} + D^{26} + D^{23} + D^{22} + D^{16} + D^{12} + D^{11} + D^{10} + D^8 + D^7 + D^5 + D^4 + D^2 + D + 1$ (appears prime, not sure)	

Analysis – CRCs are for detection ONLY.

- $P_u \triangleq$ **undetected** error probability $P_u < 2^{k-n} \cdot (\bar{P}_b)^4$; $s = 0$ for wrong codeword.

High $\bar{P}_b$ if inner code fails

Name	$P_u/(\bar{P}_b)^4$	$1 - P_u/(\bar{P}_b)^4$	Reliability	
CRC-7	2^{-7}	.99221875	2 nines	$\{$*voice reliability*$)$
CRC-8	2^{-8}	.99609375	3 nines	
CRC-12	2^{-12}	.999755859375	4 nines	$\{$*video reliability*
CRC-16 USA	2^{-16}	0.999984741210938	5 nines	$\{$*core network reliability*$)$
CRC-16 Euro	2^{-16}	0.999984741210938	5 nines	
CRC-24	2^{-24}	0.999999940395355	7 nines	$\{$*critical reliability*
CRC-32	2^{-32}	0.999999999767169	9 nines	$\{$*storage*

- These are link-layer reliabilities - $\bar{P}_b$ could be high within a CRC codeword if large$-N$ inner-code fails,
  - but still, even if $\bar{P}_b$=.1, these get very low.
- TCP-IP and higher-level session/application CRC checks (possibly using RS codes for detection) would create super reliability with "once in a century" level failures.
- These are cyclic codes so use earlier simple generators and receiver-syndrome calculation circuits.

Retransmission

- **Automatic Repeat Request (ARQ)**: If the CRC detects an error, resend the codeword.
- ARQ requires a mechanism for acknowledgment (back-channel) or ACK/NAK.
  - The NAK returns upon the receiver's non-zero CRC syndrome calculation.
  - P_c is the correct receipt probability (syndrome is zero).

$$\mathbb{E}[L_{retrans}] = \sum_{l=1}^{\infty} l \cdot P_c \cdot (1 - P_c)^l = \frac{1}{P_c}$$

- Throughput = $\left(\frac{k}{n}\right) \cdot P_c \cdot R$ bps

- Throughput represents the "real data rate" with code redundancy and retransmission accounted.
  - Throughput assumes infinite buffer delay is possible.

- There are entire courses in this network/queuing area, see EE384S (Bambos, Spring Q).

Hybrid ARQ (HARQ)

- **HARQ:** A cyclic code is used with both detection and correction.
  - If the correction part works, there is no need to retransmit.
  - If the detection part discovers an error, and then retransmission occurs.
  - Reed Solomon cyclic codes can split the parity bytes into those for correction and those for detection (sum is the allowed maximum P).

- HARQ with **soft decoding**
  - **Chase Decoding** – use all instances of (re-) transmitted codeword (form of diversity) to decode.
  - **Incremental Redundancy** – only retransmit additional parity bits (this is what 5G uses).

End Lecture 11

GF4 Tables

- $g(D) = 1 + D + D^2$ is a **primitive polynomial** in GF(2) on which GF(4) is based" $1 + D^3 = (1 + D) \cdot (1 + D + D^2)$=0
  - So, setting $g(D)$ =0 leads to $D^2 = 1 + D$.
  - A consequent GF4 primitive element is $\alpha = D$ and $\alpha^2 = D^2 = 1 + \mathrm{D}$; $\alpha = 1 + \mathrm{D}$ also works.

$\oplus$	0	1	D	1+D
0	0	1	D	1+D
1	1	0	1+D	D
D	D	1+D	0	1
1+D	1+D	D	1	0

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

$\otimes$	0	1	D	1+D
0	0	0	0	0
1	0	1	D	1+D
D	0	D	1+D	1
1+D	0	1+D	1	D

or (lsb first)

$\otimes$	00	10	01	11
00	00	00	00	00
10	00	10	01	11
01	00	01	11	10
11	00	11	10	01

or (lsb last)

$\otimes$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

GF8 Tables

$\oplus$	0	1	D	D^2	$1+D$	$D+D^2$	$1+D+D^2$	$1+D^2$
0	0	1	D	D^2	$1+D$	$D+D^2$	$1+D+D^2$	$1+D^2$
1	1	0	$1+D$	$1+D^2$	D	$1+D+D^2$	$D+D^2$	D^2
D	D	$1+D$	0	$D+D^2$	1	D^2	$1+D^2$	$1+D+D^2$
D^2	D^2	$1+D^2$	$D+D^2$	0	$1+D+D^2$	D	$1+D$	1
$1+D$	$1+D$	D	1	$1+D+D^2$	0	$1+D^2$	D^2	$D+D^2$
$D+D^2$	$D+D^2$	$1+D+D^2$	D^2	D	$1+D^2$	0	1	$1+D$
$1+D+D^2$	$1+D+D^2$	$D+D^2$	$1+D^2$	$1+D$	D^2	1	0	D
$1+D^2$	$1+D^2$	D^2	$1+D+D^2$	1	$D+D^2$	$1+D$	D	0

- $g(D) = 1 + D + D^3$, so $D^3 \to 1 + D$
  - Primitive element is $\alpha = D$

i	GF(8) element		
	α^i	lsb first	lsb last
$-\infty$	0	000	0
0	1	100	1
1	D	010	2
2	D^2	001	4
3	$1+D$	011	6
4	$D+D^2$	110	3
5	$1+D+D^2$	111	7
6	$1+D^2$	010	5

$\oplus$	0	1	2	4	6	3	7	5
0	0	1	2	4	6	3	7	5
1	1	0	3	5	2	7	6	4
2	2	3	0	6	4	1	5	7
4	4	5	6	0	7	2	3	1
6	6	2	1	7	0	5	4	6
3	3	7	4	2	5	0	1	3
7	7	6	5	3	4	1	0	2
5	5	4	7	1	6	3	2	0

- See Appendix B for matlab commands that will generate these tables.

Convolutional Interleaver Generator

$$x_k \longrightarrow \boxed{\begin{array}{c}\text{Interleaver}\\ \pi(k)\\ \text{period } L\\ G(D)\end{array}} \longrightarrow \tilde{x}_k$$

$$\boldsymbol{X}(D) = \sum_{m=0} [\boldsymbol{x}_{mL+L-1} \quad \cdots \quad \boldsymbol{x}_{mL+1} \quad \boldsymbol{x}_{mL}] \cdot D^m$$

$$\tilde{\boldsymbol{X}}(D) = \sum_{m=0} [\tilde{\boldsymbol{x}}_{mL+L-1} \quad \cdots \quad \tilde{\boldsymbol{x}}_{mL+1} \quad \tilde{\boldsymbol{x}}_{mL}] \cdot D^m = \boldsymbol{X}(D) \cdot G(D)$$

- $G(D)$ must be causal linear; D corresponds to a delay of one interleaver period.
  - If $G(D) = G(0)$, then block interleaver, otherwise a convolutional interleaver.
  - Subsymbols interleaved may themselves be vectors.
- A period has L subsymbols within it. D is one period, D_{ss} is one subsymbol period.
- To relate roughly to convolutional code, $D \to D_{ss}^L$, a period is L subsymbol periods.

$$\boldsymbol{X}(D_{ss}) = \left[D_{ss}^{L-1} \cdot x_{L-1}(D)\,\big|_{D=D_{ss}^L} \quad D_{ss}^{L-2} \cdot x_{L-2}(D)\,\big|_{D=D_{ss}^L} \quad \ldots\; x_0(D)\,\big|_{D=D_{ss}^L}\right]$$

- Simple Block Example

$$\pi(k) = \begin{cases} k+1 & \text{if } k = 0 \bmod 3 \\ k-1 & \text{if } k = 1 \bmod 3 \\ k & \text{if } k = 2 \bmod 3 \end{cases}$$

or in tabular form:

$\pi(k)$:	-1	1	0	2	4	3	5
k:	-1	0	1	2	3	4	5

,

which has inverse de-interleaver

$k' = \pi(k)$:	-1	0	1	2	3	4	5
$\pi^{-1}(k') = k$:	-1	1	0	2	4	3	5

$$G(D) = G^{-1}(D) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

This one is trivial