The ladybird guides

Book II

Day 4 – channel coding

Day 4: Coding

Objective:
To appreciate the need for coding
To describe the different schemes and their advantages & disadvantages
To appreciate that useful data rate is not the same as data rate

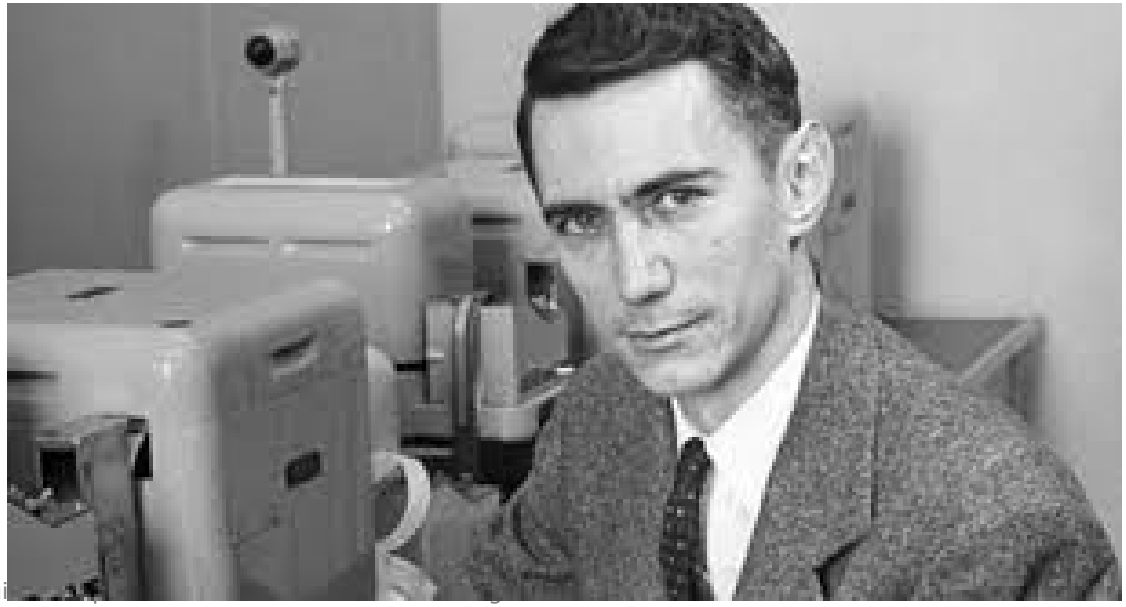
1. Simple coding techniques and examples
2. Forward Error Correction (FEC)
3. Bandwidth and data rate considerations
4. Coding gain
5. Different coding schemes usually used on the downlink
6. Useful data rate – beware of different definitions
7. Uplink coding schemes

Quiz

Who is this?

Inventor of the “the ultimate machine”
Owned 50 unicycles
Extremely good juggler
Was involved in the MIT team that used to play (and win) in Las Vegas

Age 22 wrote “most influential master’s thesis ever written” – founding the computer industry
Ten years later he wrote a famous paper - founding the digital communications industry
Most famous for discovering the equivalent “the speed of light” in communications



(https://www.youtube.com/watch?v=G5rJJgt_5mg)

Quiz

What is the best way to transmit a message along a chain of people (picture or word)?
What is the problem with analogue communication over large distances?

NOISE!! gets amplified and takes over pretty fast thanks to repeated amplification

Shannon showed us that

- All sorts of medium (music, voice, pictures) can be reduced to information bits
- The bits could be sent over a noisy channel and retrieved almost perfectly
- Then you can amplify the bits and start again sending it further
- You can improve the amount of information bits you can send using codes

But he also proved mathematically there is a limit…

$$C = B \log_2 \left(1 + \frac{S}{N}\right)$$

Once Shannon found the limit, all the engineers tried to design systems to get to it!

How does this work for spacecraft?

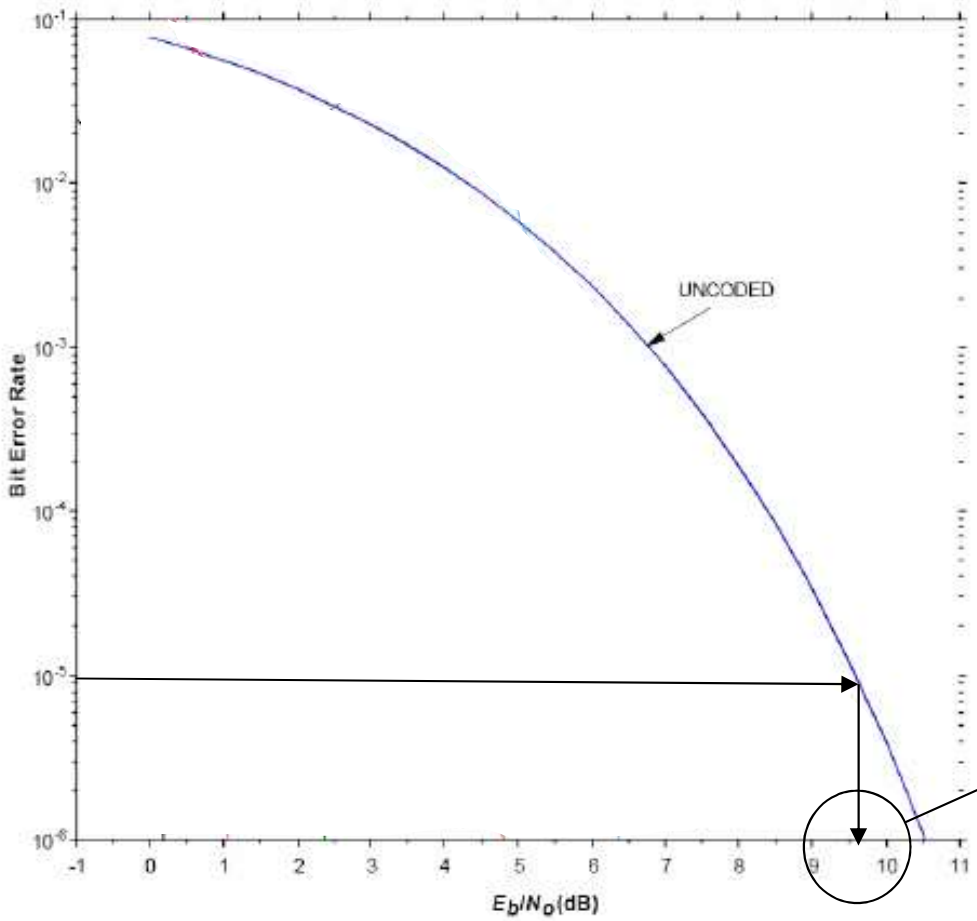
Remember the DTTL will try measure channel symbols but noise makes this difficult.

To give us the best chance of the DTTL measurement being correct we have to make sure our rock is high enough above the sea.

We use the term Eb/No (energy per information bit/noise density) to describe this height.

Eb/No and probability of error

It will never work 100% of the time. So we must first decide how much error we can accept on our link and then find the corresponding acceptable Eb/No.



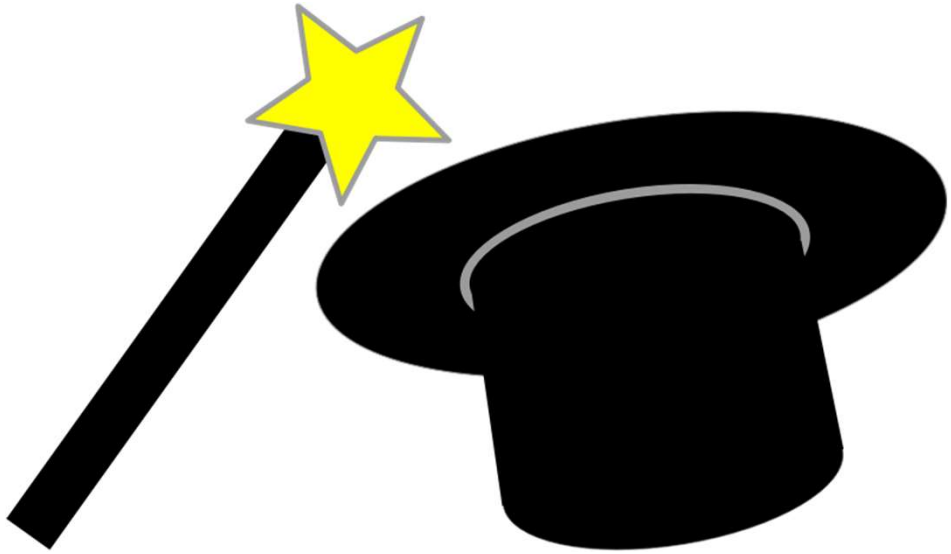
This becomes an important DESIGN goal for the link

The received power problem

$$\frac{Eb}{No} = \frac{\text{Power of received signal}}{\text{Information rate}} \times \frac{\text{Bandwidth}}{\text{Power of Noise}}$$

Received power in space communication systems is very, very low and bandwidth is limited. Hence getting an Eb/No that is high enough for the DTTL to detect our symbols correctly is challenging..

What can we do? Everything in the equation above seems fixed…



We need a bit of magic

A first bit of magic

Consider two messages from the spacecraft

1) Switch A is open
2) Switch A is closed

Quiz: What is the most bandwidth efficient way of passing the message between the receiver and transmitter?

If both the receiver and the transmitter know the same information it does not have to be transmitted.

I.e. If bit 20 = 1 then Switch A is open
If bit 20 = 0 then Switch A is closed

This is the principle of the shared operational database and is the biggest contribution to improving the information rate we have!

Another bit of magic called error detection

Now let's introduce noise into the system. Sometimes a bit in the message will be inverted. What can we do to protect ourselves?

Send it twice!

I.e. If bit 20 = 1 and bit 21 = 1 then Switch A is open
If bit 20 = 0 and bit 21 = 0 then Switch A is closed
If bit 20 = 1 and bit 21 = 0 then we have an error
If bit 20 = 0 and bit 21 = 1 then we have an error

This is error detection!

A bit of magic called FEC

Let's say we absolutely need this information even if sometimes a bit in the message is inverted. What can we do to ensure this now?

Send it three times!

I.e. If at least 2 out of 3 from bit 20,21,22 are 1 then Switch A is open
If at least 2 out of 3 from bit 20,21,22 are 0 the Switch A is closed

Plus we can safely guess that the 1 out of 3 bits has flipped and can be corrected.

This is forward error correction (FEC).

In principle that is everything we need to know about coding.

Either adding extra bits to the message so we can detect or correct error in the message

OR

Combining the bits in the message with each other to create more bits so we can detect or correct errors in the message.

Basic types of coding schemes

Either adding extra bits to the message so we can detect or correct error in the message

Linear block codes
BCH
Reed-Solomon
Hamming
LDPC

Data Blocks

OK/NOK

Combining the bits in the message with each other to create more bits so we can detect or correct errors in the message.

Convolutional codes
Convolutional
Turbo codes
SCCC

Bit Streams

Best Guess

Concatenated

Coding starts with Reed Solomon then the result is coded with Convolutional

Impact of Coding

Note coding schemes work by reducing the impact of noise at the receiver i.e. they make the system more tolerant to noise.

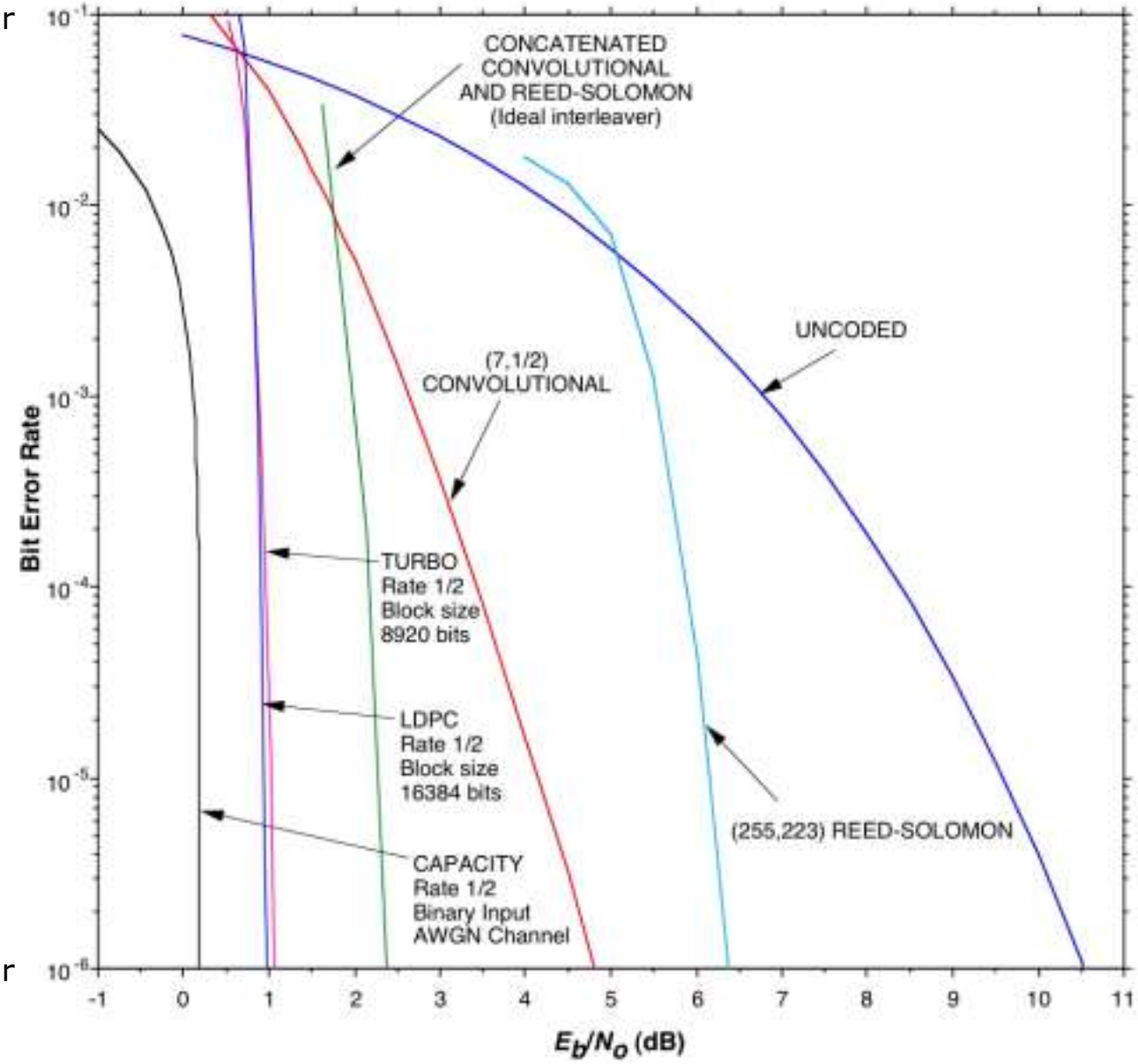
However they also introduce more bits overall so the available energy per information bit is reduced.

Therefore it is a balance, if we accept a higher bit error rate then the benefits of the coding scheme are reduced while the impact of having less energy per information bit is not. Sometimes it can even go negative i.e. better to have no coding!

As a rule of thumb, the more effective the scheme the more redundancy bits it introduces..

Eb/No versus BER for different coding schemes

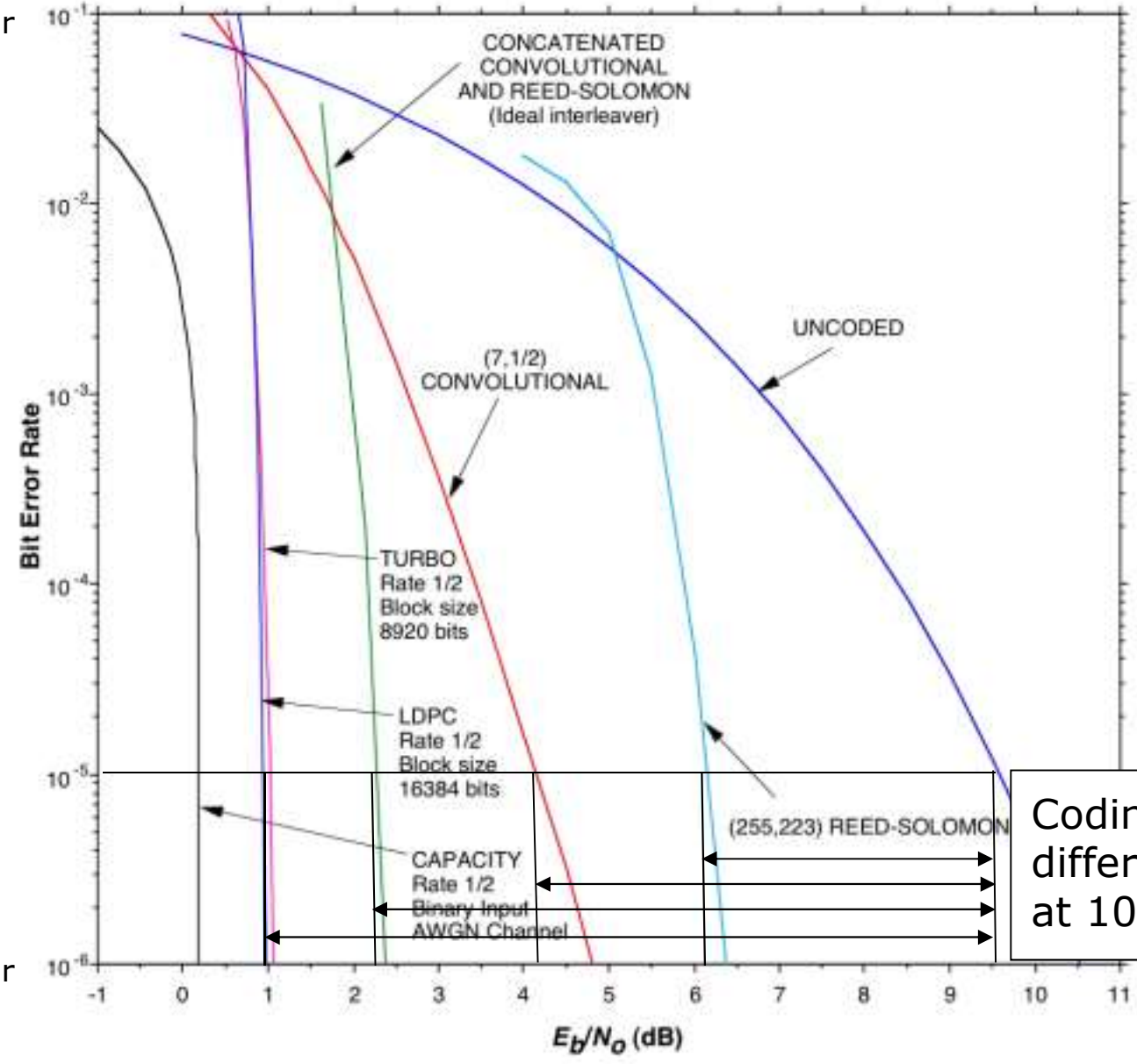
High error rate



Low error rate

How do we calculate the advantage?

High error rate



Low error rate

For convenience, we represent the benefit of this coding as a gain compared to an un-encoded message.

It is called CODING GAIN.

Coding Gain is simply the difference between the Eb/No we would of required for an uncoded data stream and that required for the coded data stream.

Coding gains for different schemes at 10^{-5} BER

WARNING BER!

There is no free lunch!

Remember all the bits (information and redundant) have to be passed to the modulator.

Hence coding increases the number of symbols and therefore bandwidth usage.

If we cannot afford that we have to decrease the information rate by the corresponding amount to keep the number of symbols the same.

The decrease factor is called the Code Rate e.g. 0.5

For a bandwidth constrained system we accept less data throughput for a lower Eb/No threshold

For a data throughput constrained system we accept more bandwidth usage for a lower Eb/No threshold

RECAP: Bandwidth calculations without coding

Ideal bandwidth after modulation = Symbol rate

but this bandwidth is needed to carry the symbols only… in order to reduce to -20dB power by the next user we have to apply the roll off…

Occupied bandwidth = (1 + roll off) x Symbol rate

Note if we apply channel coding (see later) we increase the number of symbols which has to be taken into account here…

Without channel coding we can say

$$\text{Symbol rate} = \frac{\text{information rate}}{\text{number of bits per symbol}}$$

Bandwidth calculations with coding

Ideal bandwidth after modulation = Symbol rate

but this bandwidth is needed to carry the symbols only… in order to reduce to -20dB power by the next user we have to apply the roll off…

Occupied bandwidth = (1 + roll off) x Symbol rate

Note if we apply channel coding (see later) we increase the number of symbols which has to be taken into account here…

With channel coding we can say

$$\text{Symbol rate} = \frac{\text{information rate}}{\text{number of bits per symbol}} \times \frac{1}{\text{coding rate}}$$

With QPSK and convolutional coding (1/2) occupied bandwidth is about 1.35 x data rate

Why?

Bandwidth calculations

The general formula for calculating bandwidth has to be modified from

$$\text{Actual bandwidth required} = \frac{(1 + \text{roll off}) \times \text{information rate}}{\text{Number of bits per symbol}}$$

to..

$$\text{Actual bandwidth required} = \frac{(1 + \text{roll off}) \times \text{information rate}}{\text{Number of bits per symbol}} \times \frac{1}{\text{coding rate}}$$

Bandwidth / information rate impact

Coding scheme	Bandwidth relative to uncoded channel (a)
Uncoded	1
(255, 223) Reed-Solomon only	1,14
Punctured convolutional rate 7/8	1,14
Punctured convolutional rate 5/6	1,2
(255, 223) R-S and punctured convolutional rate 7/8	1,31
Punctured convolutional rate 3/4	1,33
(255, 223) R-S and punctured convolutional rate 5/6	1,37
Punctured convolutional rate 2/3	1,5
(255, 223) R-S and punctured convolutional rate 3/4	1,52
(255, 223) R-S and punctured convolutional rate 2/3	1,71
Turbo code rate 1/2 (information block length = 8920 bits)	2
Basic convolutional k=7, rate 1/2	2
(255, 223) R-S and basic convolutional rate 1/2	2,28
Turbo code rate 1/4 (information block length = 8920 bits)	4

Coding - quiz

You are a spacecraft communication system designer. Your mission is marginal in terms of signal to noise ratio. You been given the following two options to improve the situation.

1. Implementation of convolutional coding in the telemetry reducing your required Eb/No by 7dB

2. Switch to an bigger (more expensive) ground station antenna and gain 5 dB on the received signal power.

Which would you choose and why?

This is a trick question because it depends

1. If you are not bandwidth limited option 1 is better. We can trade extra bandwidth for a lower Eb/No requirement.

2. If we are bandwidth limited than option 1 will mean reducing the useful data rate by 50%. Hence option 2 is better.

Basic types of coding schemes

Either adding extra bits to the message so we can detect or correct error in the message

Linear block codes
BCH
Reed-Solomon
Hamming
LDPC

Data Blocks

OK/NOK

Combining the bits in the message with each other to create more bits so we can detect or correct errors in the message.

Convolutional codes
Convolutional
Turbo codes
SCCC

Bit Streams

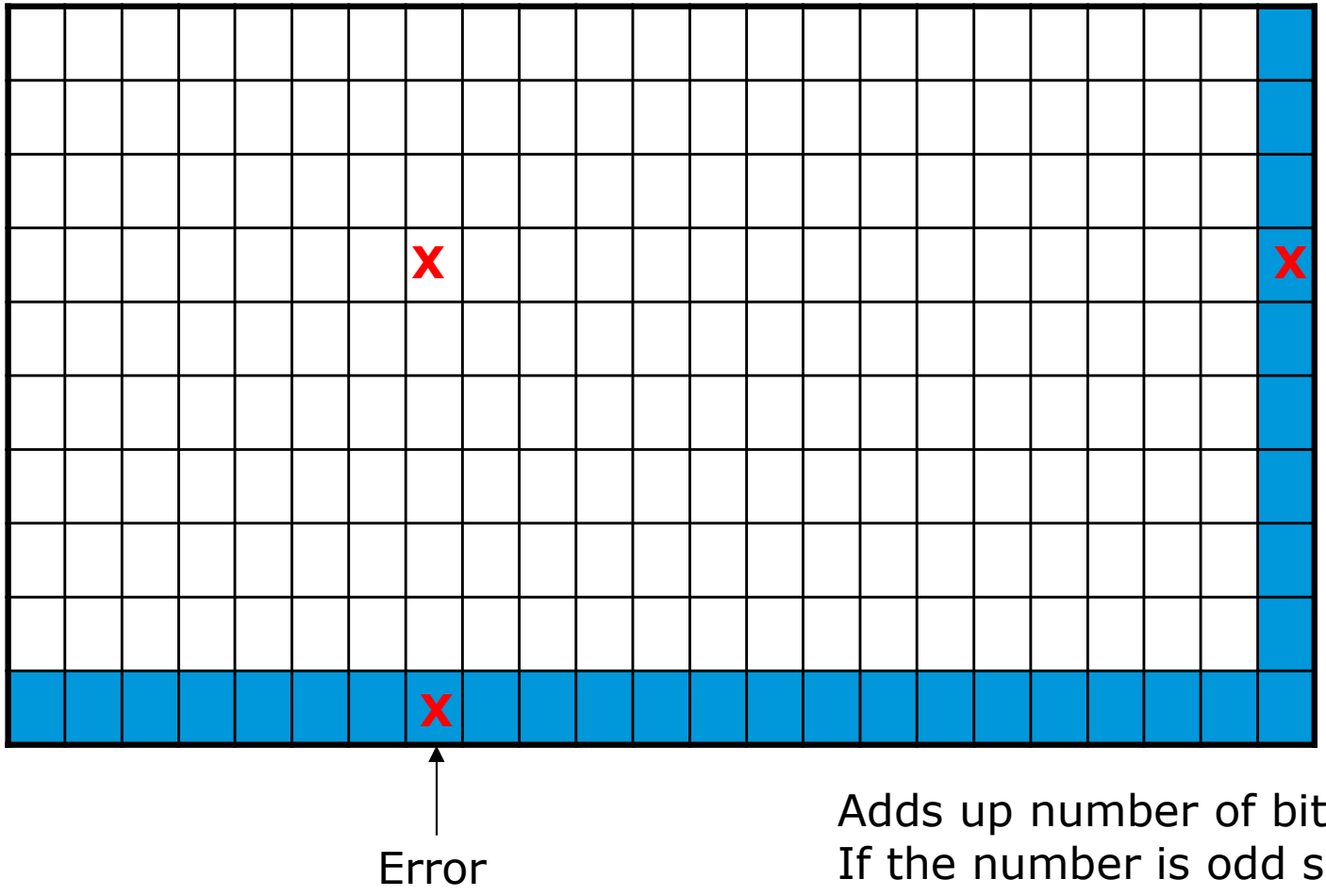
Best Guess

Concatenated

Coding starts with Reed Solomon then the result is coded with Convolutional

Block Codes

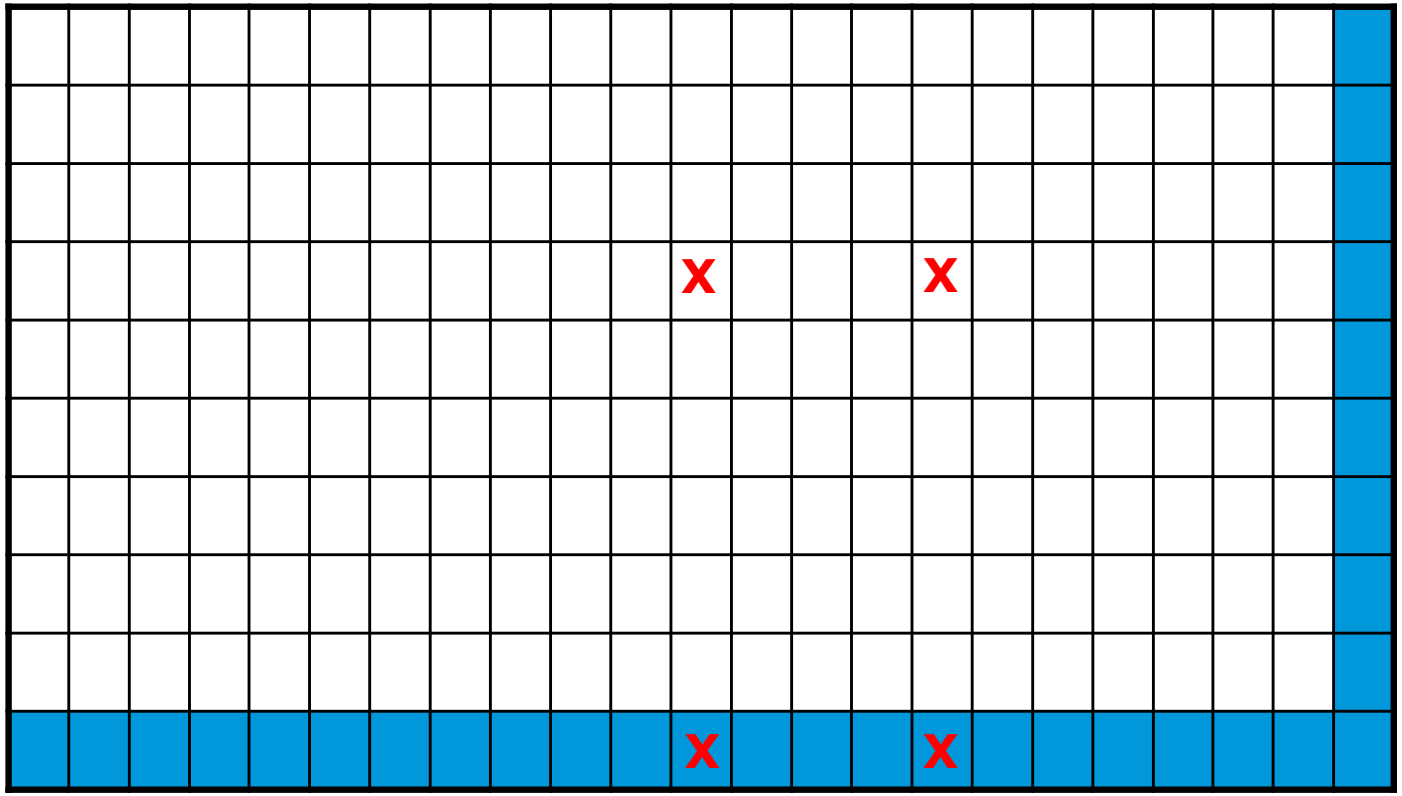
Parity bits in blue
Message bits in white



Adds up number of bits with the number 1
If the number is odd set the parity one way
otherwise set it the other

1011000110011 = parity bit is set to 1
0101001101010 = parity bit is set to 0

Error bursts and Code blocks



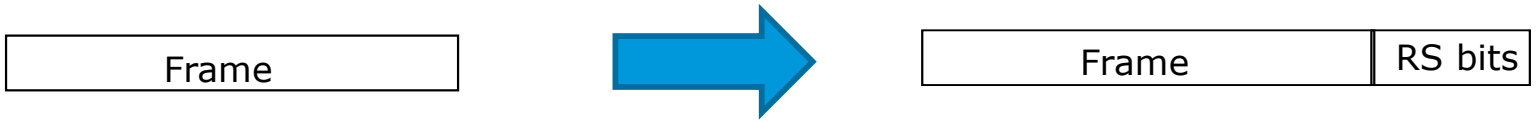
Block codes deal with RUNs or BURSTS of errors very effectively.
Too many errors result in correction not being possible but this failure can be flagged.

This detection property can be very useful. If it passes the decoding stage then there is no need to implement checksums for later checking.

R-S coding

Reed-Solomon coding is a block code (extension of the parity matrix idea).

A trailer is added to the original transfer frame data containing the parity bits. We call this systematic as the original bits are sent in the clear.



The code is usually described by this relationship

Output clock size, Transfer Frame size, Interleaving depth (see later).

There is an option on how much parity information is added called E (error correction capability), either 16 or 8.

If E= 16 (normally used) then the code will be

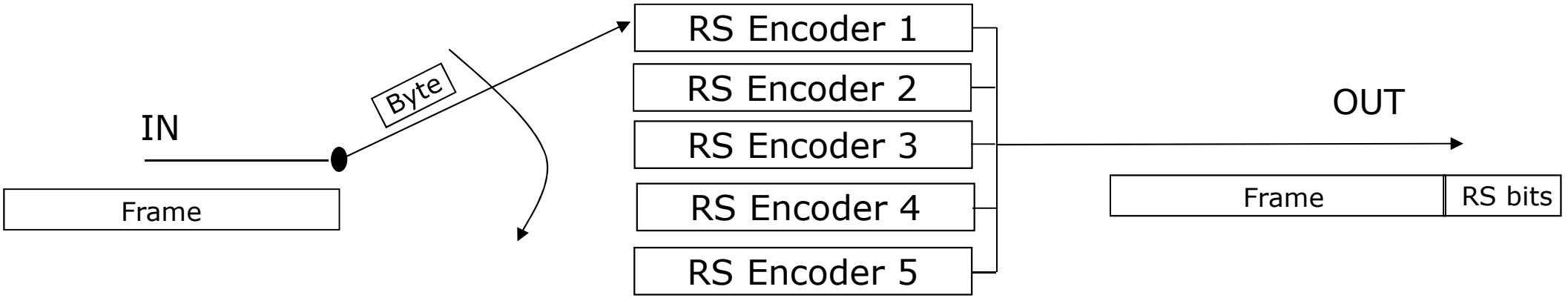
255, 223, I

The coding rate is therefore 223/255 = 0.8745 (i.e. 14% expansion)

Interleaving

Interleaving works by splitting the incoming data up every byte (called a RS symbol) and passing them to an array of RS encoders.

The number of encoders in the array is called the interleave depth (I). Therefore the larger I, the longer the final coded frame.



So for example if we have R-S coding (255, 223) and I = 5.

Then the coded frame after R-S coding will be 223 x 5 bytes (1115 bytes) of interleaved data with 32 x 5 bytes (160 bytes) attached of interleaved parity bytes.

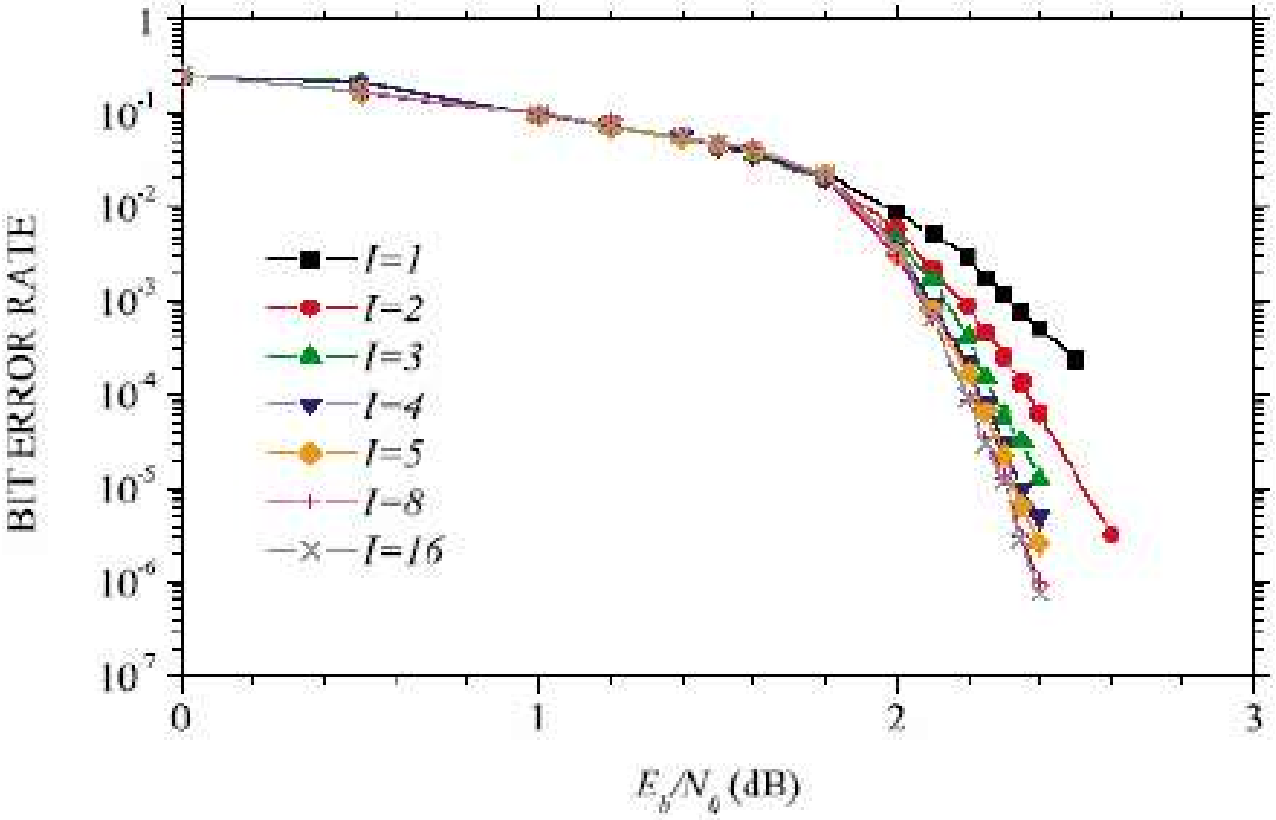
Hence the final frame is 1275 bytes long (plus 4 bytes for a ASM, see day 5).

Interleaving

The advantage of interleaving is that bursts of errors in the incoming stream are split up.

The RS decoder is able to handle single errors better than groups (think back to the parity block diagram).

Hence interleaving helps the RS coding reduce the effect of noise even more.

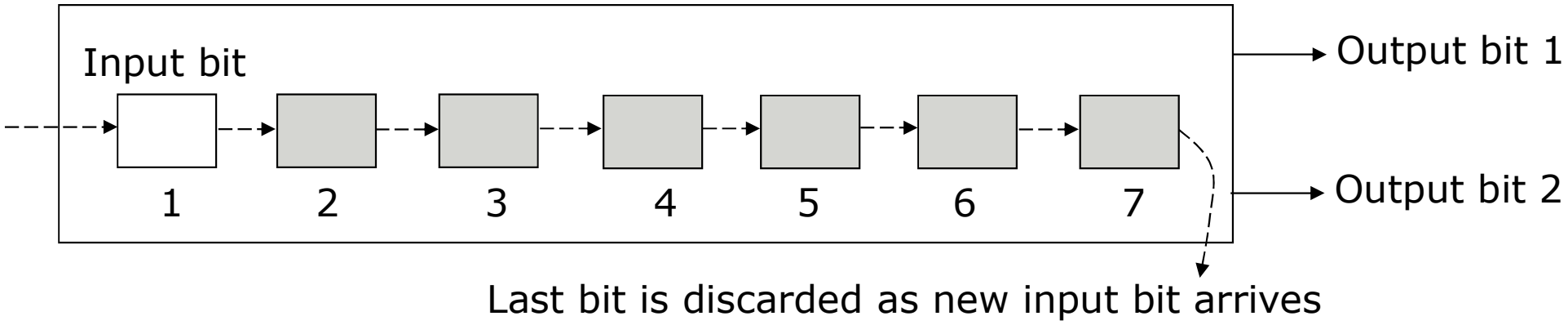


Convolutional Coding

Convolutional coding is NOT a block code. It works by processing a bit stream.

Think of it as a machine constantly eating input bits at one end and spitting out multiple output bits at the other

Initially all the registers are filled with zeros. As the input bit arrives, it pushes all the other bits stored in the memory registers along by one.



It is not systematic, i.e. the original information bits are not send in the clear and it is a best guess system.

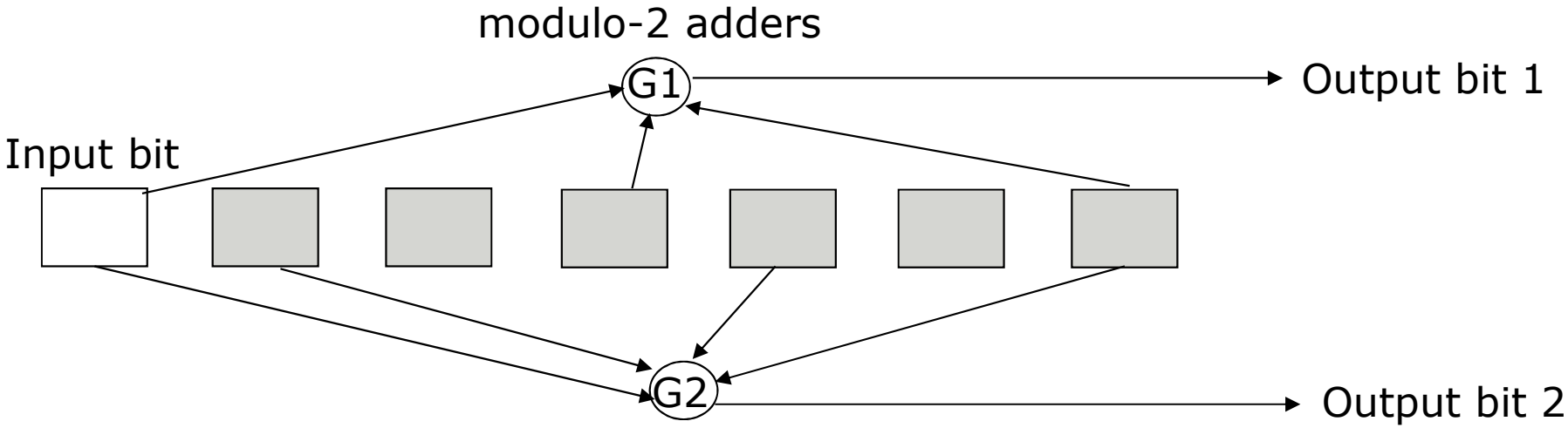
Convolutional Coding

The contents of selected registers are combined together in a number of modulo-2 adders (0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 0) producing the output bits.

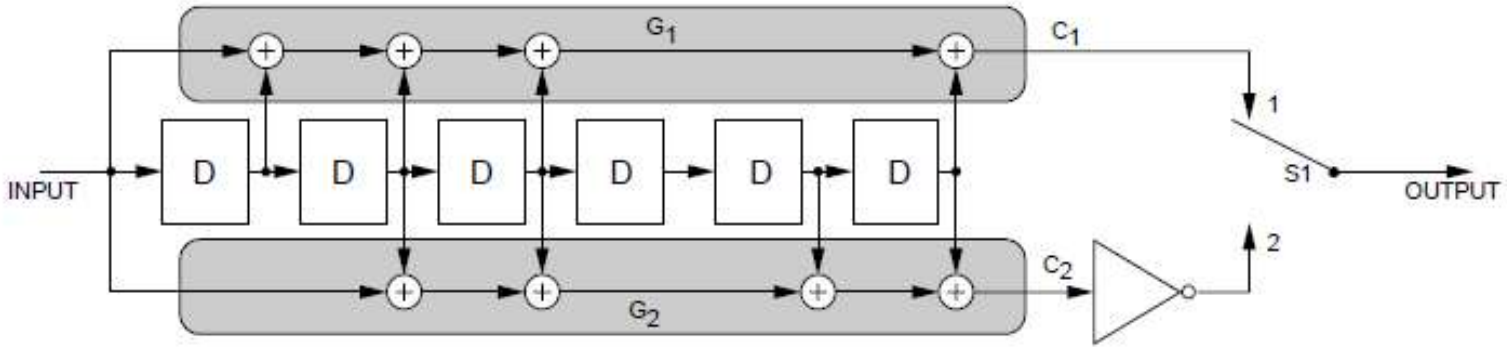
How the adders are arranged is called the generator polynomials (G1 and G2). For example below the top and bottom modulo-2 adders have the polynomials

G1 = 1,0,0,1,0,0,1 and

G2 = 1,1,0,0,1,0,1 respectively, due to their connections.



Standard CCSDS Convolutional encoder



Standard CCSDS convolutional decoder
½, k = 7, no puncturing, G1 = 1111001, G2 = 1011011

Note that there is a bit inverter for the output of G2, so the process is
insert bit, calculate C1 and output it, calculate C2, invert it and output it,
insert bit, calculate C1 and output it, calculate C2, invert it and output it… and so on.

Cassini uses a k of 15 and a rate of 1/6. This means 5 out of 6 bits are redundancy bits! This trade gave them an extra Eb/No gain of 2dB (x1.5) which was critical given the distance to Saturn.

Convolutional decoding

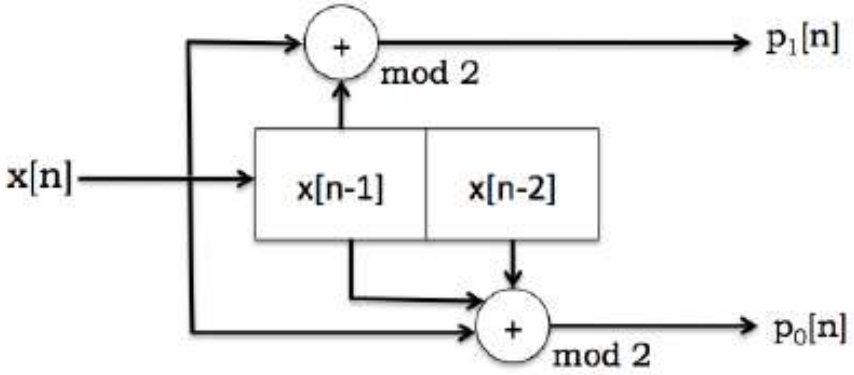


Figure 7-2: Block diagram view of convolutional coding with shift registers.

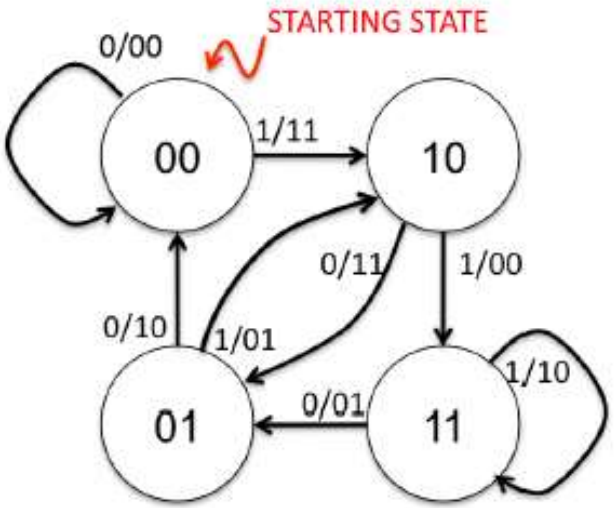


Figure 7-3: State-machine view of convolutional coding.

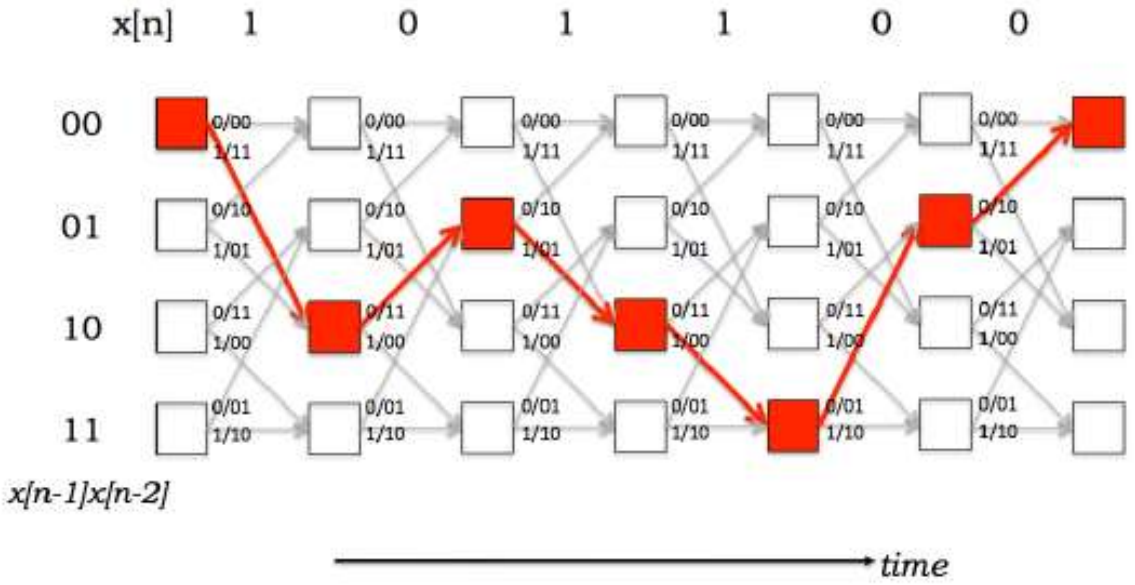
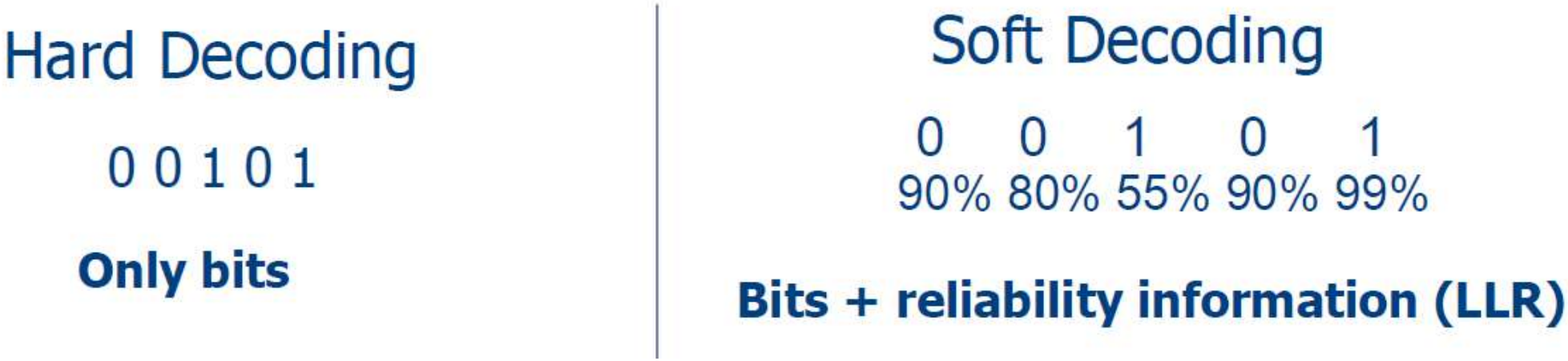


Figure 7-5: The trellis is a convenient way of viewing the decoding task and understanding the time evolution of the state machine.

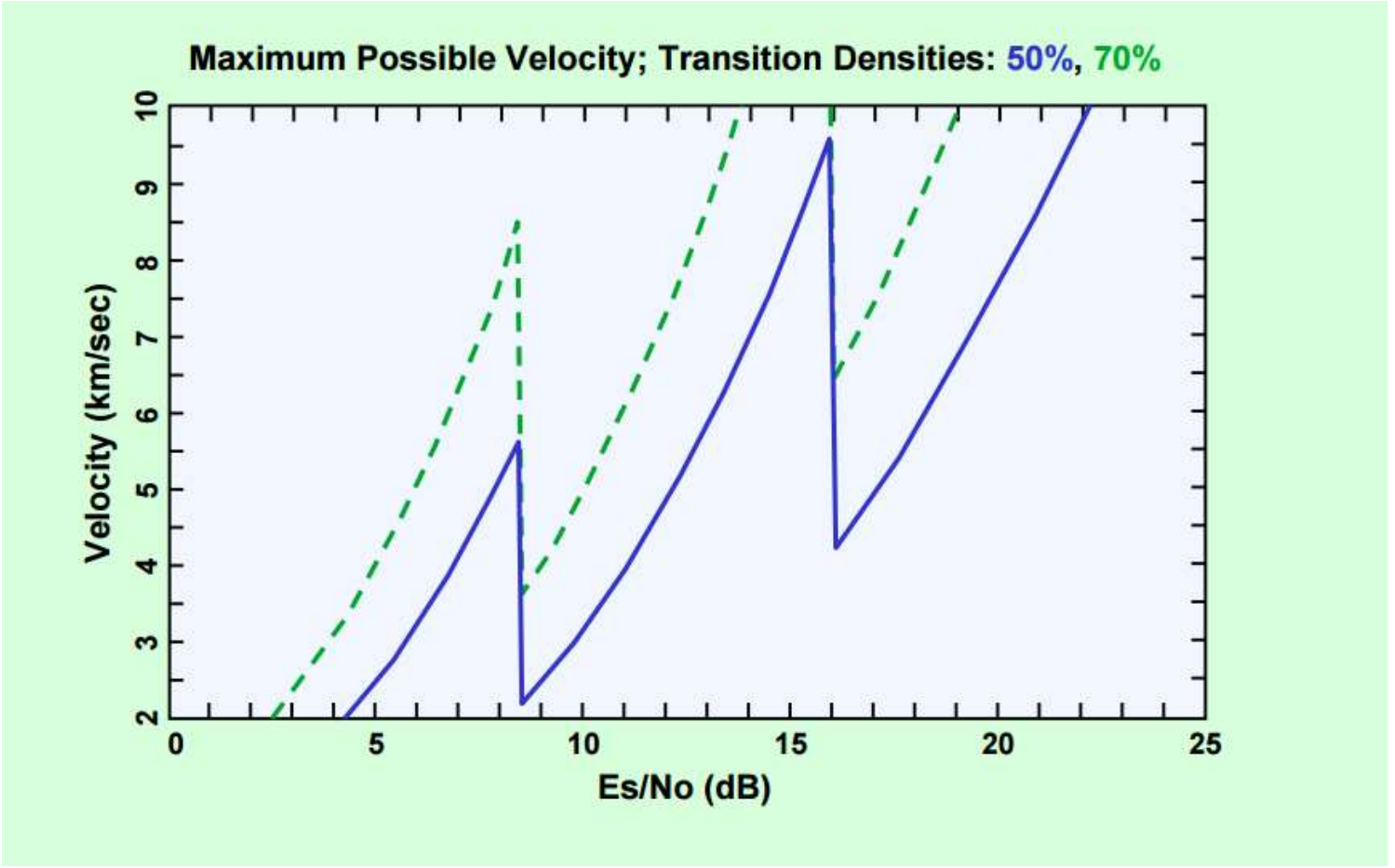
Hard and soft decoding



Reliability info comes from DDTL of each of those symbols….

Could also use for bits we know backwards?

Huygens-Cassini Story and coding



One of the proposed solutions to improve the performance of the DTTL loop on Huygens-Cassini was to insert packets full of zeros in the data stream.

Can you now explain why packets full of zeros might work?

Convolutional Coding and Puncturing

We can see that the relationship between the input and output bits is always 1 to n. However if we simply not send one of the output streams we can get an x to n relationship e.g. a code rate of 2/3.

This technique is called puncturing or "perforated".

An example puncturing pattern for a 2/3 code would be C1:1,0 and C2: 1,1.

So the output would be C1(bit 1), C2(bit 1), C2(bit 2), C1(bit 3), C2(bit 3) etc..

Obviously punctured codes are not as good at reducing the effect of noise as full codes hence we trade coding gain against more information flow.

However these codes have the advantage that it is very easy to program in software and is therefore change if the link has more or less power to due variations in distance, geometry and weather that were discussed in the challenges presentation.

Convolutional decoding

Convolutional coding is relatively easy, but decoding is not..

Decoding relies on the fact that for any given state of 6 memory registers not every possible transition of the output bits is possible.

Therefore when an illegal transition occurs the decoding algorithm knows a mistake was made and can backtrack to the most probable point it was correct and then try another route.

One can imagine this takes quite a bit of processing power.

This imbalance in processing requirements makes it perfect for use in spacecraft telemetry, where we have little resources on the spacecraft and unlimited resources on the ground.

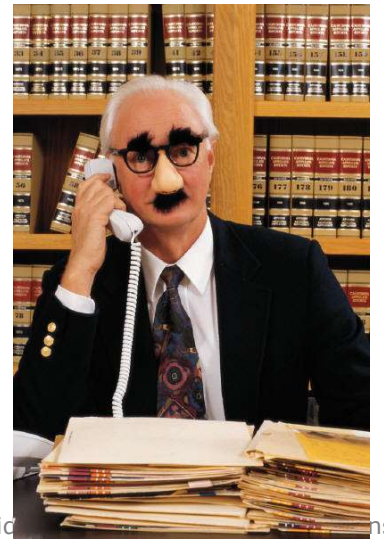
The Andrew Viterbi Story

An Italian Jew, his family escaped from Italy to the US 5 days before the start of WWII

His first job after university was working for JPL. He helped develop the first PLL which was needed for the TTC system of Explorer 1. This was the "American Sputnik" (Feb 1958) and was the spacecraft that discovered the Van Allen radiations belts

In 1962, he got a job at University of California teaching digital communications

He found standard convolutional decoding too complex and figured out a new simplified way to teach it. This turned out to work even better than the standard way



He went to a patent lawyer to ask if it was worth patenting

He said "no it is too specialised"

The Andrew Viterbi Story

Every mobile phone in the world now runs the Viterbi algorithm!

However he did not do too bad….

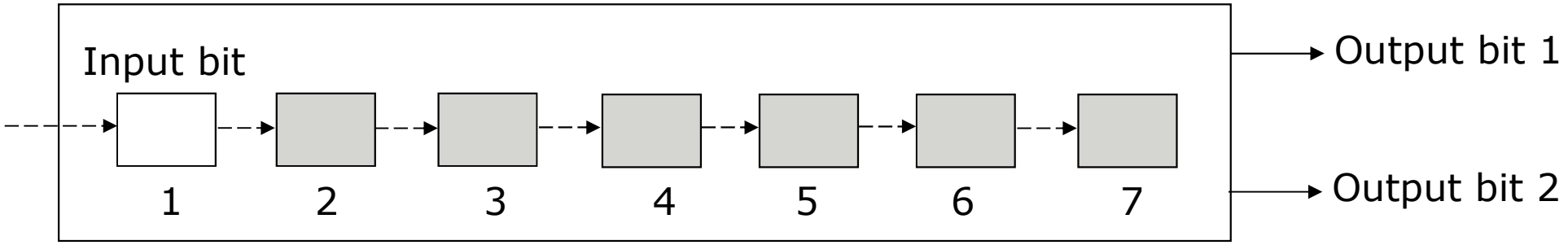
He left teaching a co-founded a series a companies specialising in spread spectrum and CDMA (code division multiple access) communication

His most famous one is Qualcomm and it turned over 26 billion dollars in 2014! The majority of the income comes from patents on 3 and 4G mobile phone technology

Explorer 1

Convolutional Coding and Error Bursts

Think of as convolutional coder as a machine constantly eating a bit at one end and spitting out two bits at the other



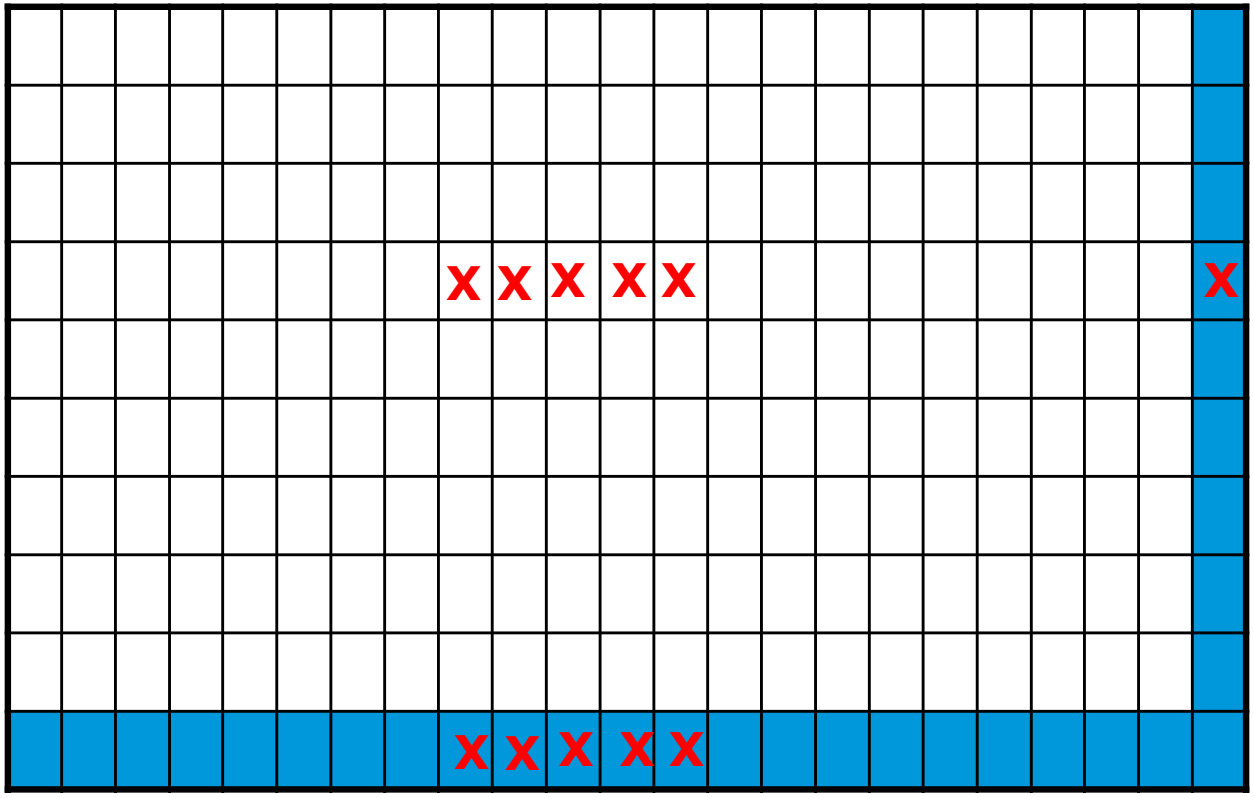
QUIZ: Convolutional encoders tend to produce error bursts if something goes wrong. Why?

Because the multiple output bits are connected to multiple input bits

Concatenated Coding

Concatenated coding is simply R-S coding followed by convolutional coding

The advantage is that during decoding, the R-S will be decoded last. Hence it can deal effectively with any bursts of errors caused by the convolutional encoder.



QUIZ: If R-S causes an expansion of 14% what is the coding rate of concatenated coding?

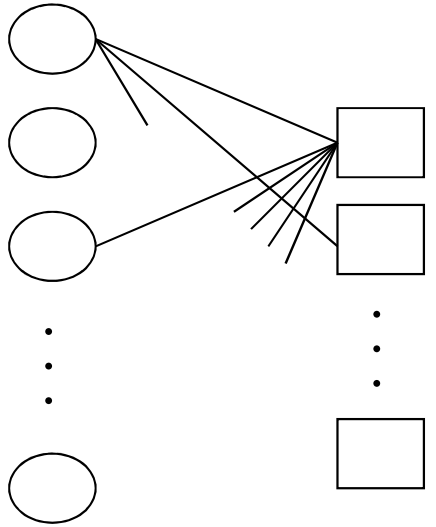
New Coding Schemes – LDPC

Low-Density Parity-Check (LDPC) codes are binary block codes.

Robert G. Gallager developed the LDPC concept in his doctoral dissertation at MIT in 1960. The thesis won an IEEE IT Society Golden-Jubilee Paper Award in 1998!

Parity-Check bits are computed from information bits using a sparse matrix with a (very small) number "1"s per column/row i.e. low density. The matrix shows how each check bit derives from its state "some" information bits, an information bit may contribute to many check bits. This can also be represented by nodes of a bipartite graph.

$$H = \begin{matrix} 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{matrix}$$



Allows simple, fast (iterative) decoding (compared to turbo)

Courtesy of Gian Paolo Calzolari ESA/ESOC

New Coding Schemes – SCCC

Serially Concatenated Convolutional Code apply to the information bits in sequence

- A Convolutional encoder (outer code);
- An interleaver;
- A Convolutional encoder (inner code);

Note for this to work the input stream is sliced into blocks of given length to be sent to the input of the encoder (after ASM insertion, see later)

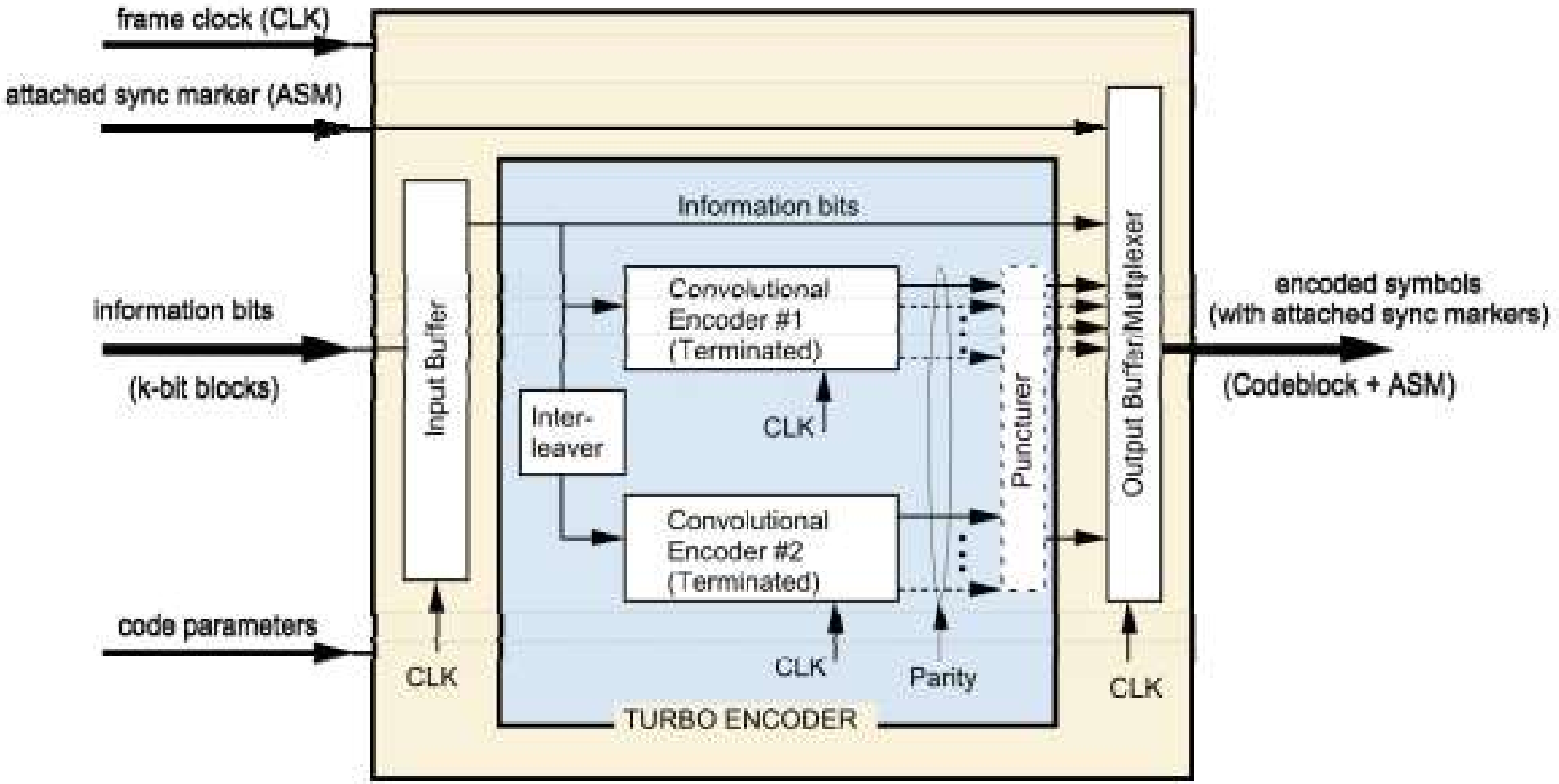
The output is non systematic (information bits not send in the clear) and is best guess (as it using convolutional encoding).

Courtesy of Gian Paolo Calzolari ESA/ESOC

New Coding Schemes – Turbo

This generates parity bits from two parallel convolutional encoders. However it is a systematic code as it outputs the original information bits plus the two sets of parity bits

Magic is in the interleaver used to generate the input for the second encoder.



Courtesy of Gian Paolo Calzolari ESA/ESOC

Types of telemetry coding schemes

Type	Name	Comments
Linear Block Code	Reed-Solomon LDPC	RS can be used with convolutional in concatenated coding LDPC will be used by EUCLID
Cyclic Code	Cyclic Redundancy Check	Used for packets and frames FECs, see protocol session
Trellis	Convolutional	Used with RS in concatenated coding
Mixed with interleaving	Concatenated	RS followed by convolutional
Mixed, recursive with interleaving	Turbo	Will be used on Bepi. "Could" have been used on ROSETTA

Telemetry Coding Schemes

Coding scheme	Bandwidth relative to uncoded channel [a]	Coding gain [b] for frame length = 8920 bits (dB)	
		FER = 10^{-4}	FER = 10^{-6}
Uncoded	1	0 [c]	0 [d]
(255, 223) Reed-Solomon only	1,14	5.6	6.3
Punctured convolutional rate 7/8	1,14	4.0	4.1
Punctured convolutional rate 5/6	1,2	4,9 [e]	5.1
(255, 223) R-S and punctured convolutional rate 7/8	1,31	7.0	7.9
Punctured convolutional rate 3/4	1,33	5.5	5.5
(255, 223) R-S and punctured convolutional rate 5/6	1,37	7.5	8.5
Punctured convolutional rate 2/3	1,5	5.9-6.0	6.0-6.1
(255, 223) R-S and punctured convolutional rate 3/4	1,52	8.2-8.3	9.2-9.3
(255, 223) R-S and punctured convolutional rate 2/3	1,71	8.7-8.8	9.7-9.8
Turbo code rate 1/2 (information block length = 8920 bits)	2	10.9	11.3
Basic convolutional k=7, rate 1/2	2	6.3-6.4	6.5-6.6
(255, 223) R-S and basic convolutional rate 1/2	2,28	9.3-9.5	10.3-10.5
Turbo code rate 1/4 (information block length = 8920 bits)	4	11.7	12.5

No coding, FER = 10^{-4} :
Eb/No = 11.9 dB

No coding, FER = 10^{-6} :
Eb/No = 13 dB

Note use of FER not BER. This is very important

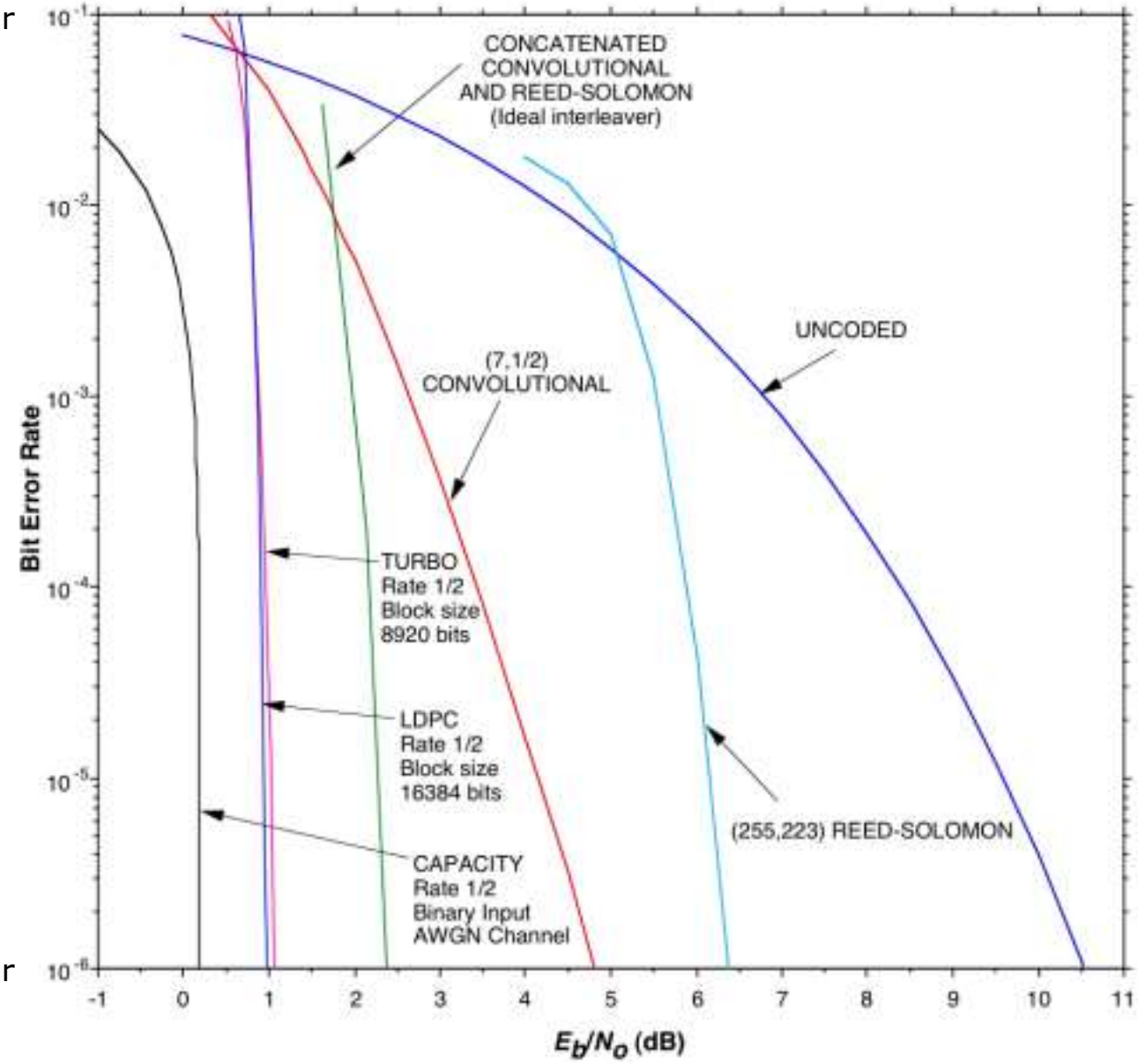
A typical uncoded telemetry frame is 1115 bytes long

To achieve a Frame Error Rate (FER) of 10^{-5} you need a Eb/No that is 2.9dB better than a BER of 10^{-5}

Remember the same effect can happen with compression

Coding - recap

High error rate



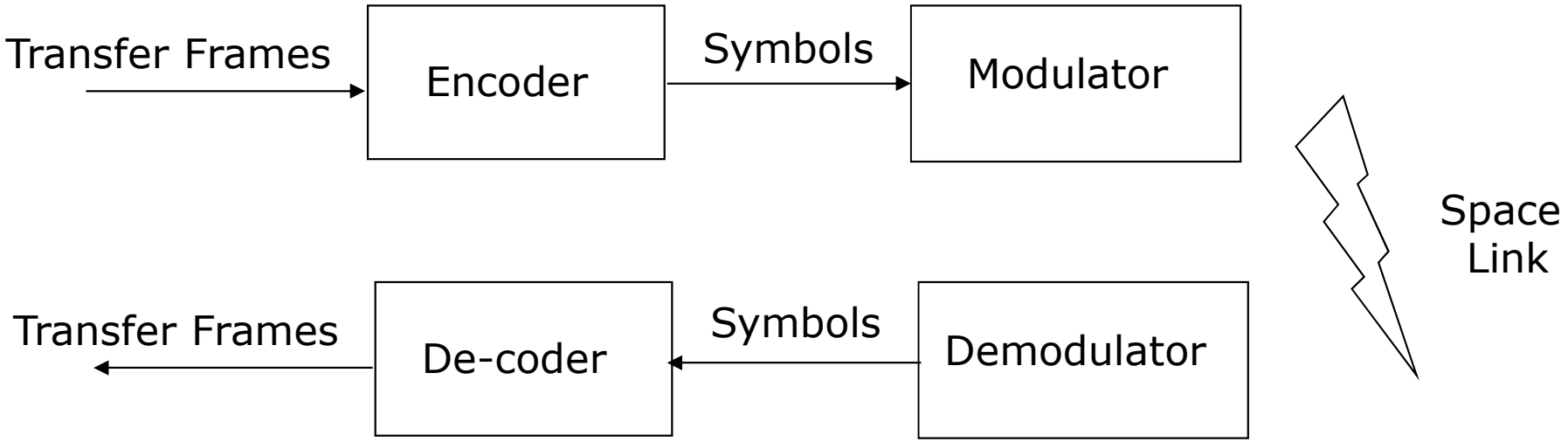
Low error rate

This diagram is very important

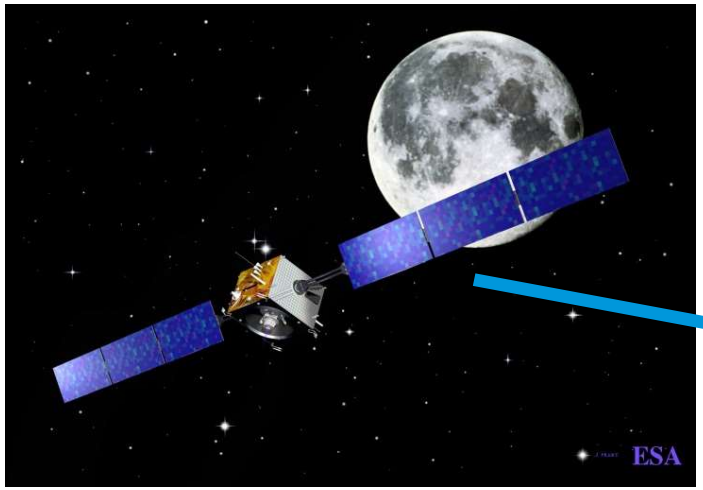
You can work out what Eb/No your system needs to achieve

And therefore the coding gain for the scheme chosen.

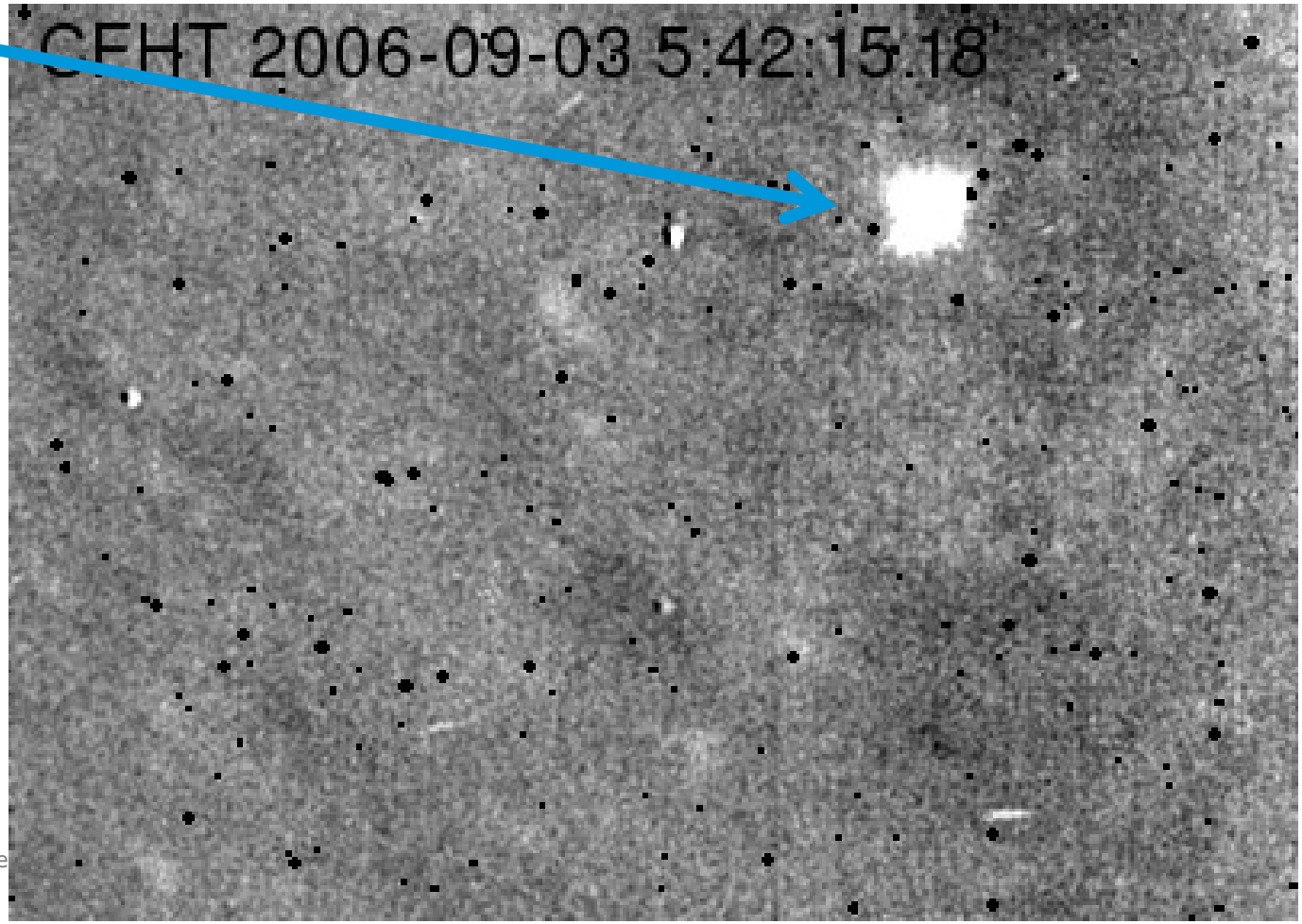
Coding - recap



Operational Experience



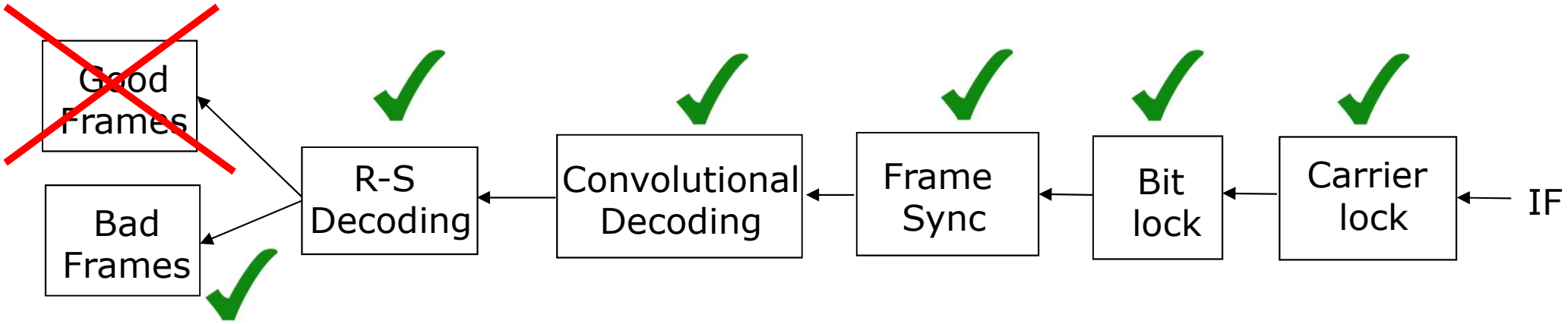
SMART-1: Launched 27th Sept 2003
Technology Demonstration Mission, Ion engine OCS etc.
Planned crash into the moon on 3rd Sept 2006



Operational Experience

3 hours after launch sudden loss of telemetry. This followed a command to change to the high rate telemetry mode.

The FCT ran the generic procedure which showed that the spectrum from the satellite looked good, they also had carrier, bit and frame lock.



They found a procedure called "reset of TMTC card" and ran it. Good telemetry was restored. Unfortunately this procedure also reset the on-board time – bad news. Radiation induced SEU was suspected as the cause.

The next day it happened again. So now they decided to switch off Reed Solomon checking at the ground station. Good telemetry was restored again!

Courtesy of Rick Blake, Sascha Ricken & Rolf Maarschalkerweerd, ESA/ESOC

Operational Experience

Then the mission control centre crashed!

Since Reed Solomon was implemented there was no checksum at frame level. With no Reed Solomon checking bad data was getting into the system. This was especially critical if the CLCW became corrupt as it could result in the retransmission of commands, see Day 5.

An interim solution: by switching backwards and forwards between the telemetry modes it was found that occasionally the Reed Solomon coding started working again. But since there was only a 1 in 13 chance of recovery this could take hours.

A more permanent solution was eventually found, see Day 5.

Why was this problem not found during ground testing?

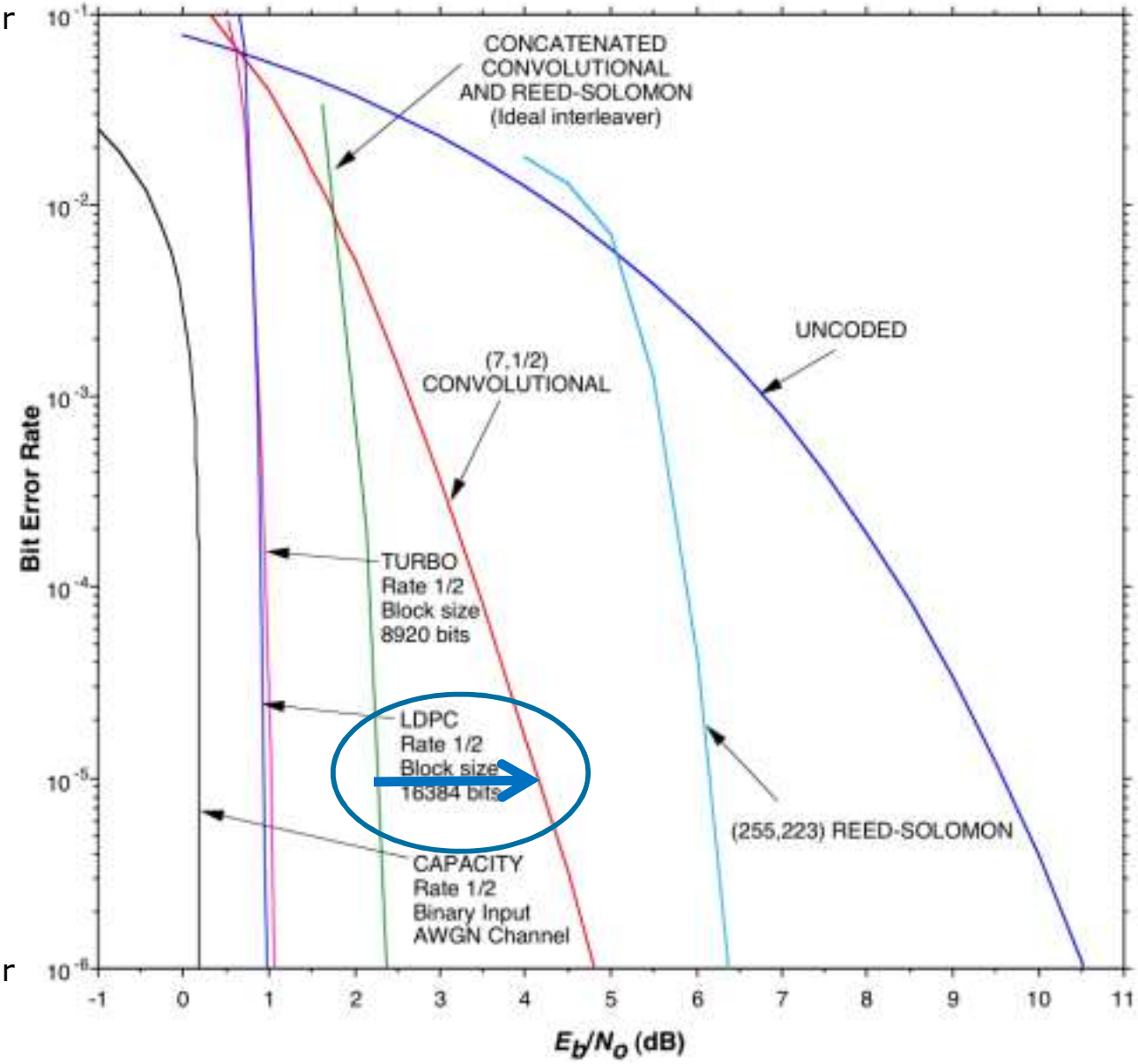
"Reed Solomon failure occurred at regular intervals during the radio frequency compatibility test in ESOC. Telemetry files were sent to SSC and ESTEC for analysis. The problem was identified in the suitcase and the RFCT was completed"

QUIZ: What is the operational lesson?

Courtesy of Rick Blake, Sascha Ricken & Rolf Maarschalkerweerd, ESA/ESOC

SMART-1 recap

High error rate



Low error rate

SMART-1 was using concatenated coding

It then effectively lost Reed Solomon coding

So it changed from concatenated to convolutional coding

It lost 2.8 dB in terms of Eb/No

It also lost the ability to stamp frames as good or bad since convolutional is best guess

Result was bad frames got into the system

Operational Experience

Impact of RS encoding not working for Smart-1:

- It causes numerous problems during LEOP because it was an intermittent problem that required manual intervention every time
- SCOS is not designed to handle bad and corrupted frames and started crashing once RS coding checks were disabled at the ground station. This forced us to implement a very expensive patch in the Smart-1 SCOS system to handle frames with errors.
- Smart-1 was using spare ground station capacity. This included 5 m, 15 m and 35 m dishes. The RS gain drop forced us to use bigger dishes that were more expensive and less available.
- Once in the moon, we had to constrain the data rate depending on which antenna was available. This complicated the ground station booking and the amount of science data that could be downlinked.

QUIZ: What could we have done had convolutional coding failed?

QUIZ for Ladybird Oners: Why is setting the on-board clock bad news?

Courtesy of Octavio Camino, ESA/ESOC

VCM and ACM

Coding trade less information rate (or more bandwidth usage) for a lower Eb/No

Note the similarity with the discussion on Modulation e.g. going from BPSK to QPSK doubles my information rate but requires twice the signal power.

Note also from Day 1 that the signal power typically changes enormously during a pass or mission.

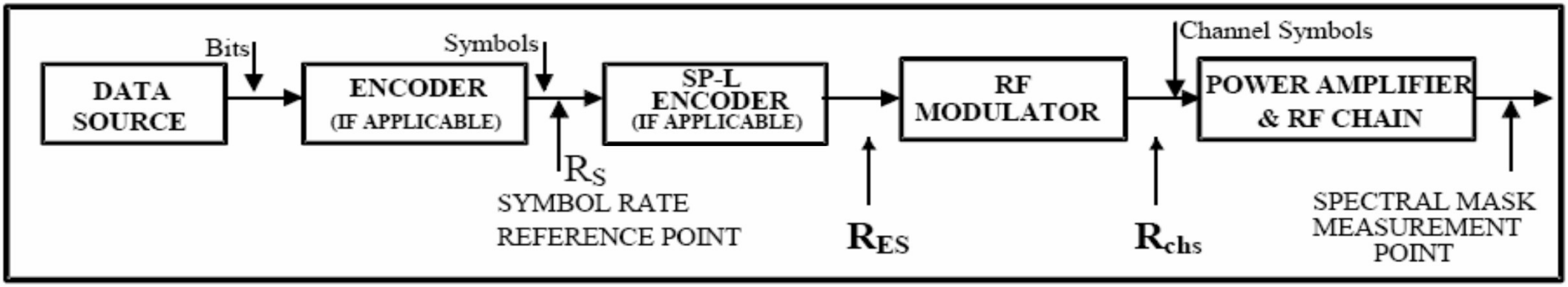
It makes sense to exploit this variation by switching between different modulation techniques and coding schemes. This allows:

- Variable Coding and Modulation (VCM) mode: schemes follow a predetermined schedule
- Adaptive Coding and Modulation (ACM) mode: uses a feedback channel to dynamically adjust the schemes according to real-time performance measurements

SCCC is particularly suited to such modes as it supports a wide variety of coding rates.

Courtesy of Gian Paolo Calzolari ESA/ESOC

Data rate definition nightmares



We should remember some definitions and make sure everyone sticks to them

- Data rate is the input to the first coder (remember it is not useful information rate due to the overheads).
- Therefore Data rate = Transfer Frames per sec x Transfer frame size

- Symbol rate is the input rate to the Modulator (expect when SP-L encoding is used)
- Therefore symbol rate = Data rate x code rate

- Channel symbol rate is the output from the modulator
- Therefore Symbol rate / number of bits per symbol (except when SP-L encoding is used)

Coding advise

Coding is a trade off between

1) Information rate and Eb/No needed
2) Bandwidth usage and Eb/No needed

It is different for telemetry and telecommand mainly due to on-board processor limitations.

Use FER on coded data (not BER) as your starting point. Watch out for similar magnification effects if data compression is used.

Be very careful to define what you are using for information rate, data rate, symbol rate. Make sure everyone else is using the same.

VCM and ACM will (hopefully) be stars of the future.

The difference between uplink and downlink

The telemetry coding schemes are chosen so that they require much less processing power to code than to decode.

Why? Let's go back to our challenges….

PROPERTY	SPACECRAFT	GROUND STATIONS
Processing Power	50-83 MIPS	Practically unlimited

However telecommand schemes have exactly the opposite problem. Unfortunately they have to be treated differently and this leads to an asymmetric link.

Note that telecommands are usually decoded by the TC decoder which is an ASIC or FPGA not the main computer. Hence it has often has even less processing power than these figures.

Uplink Coding

The Telecommand data rate is usually much lower than the telemetry data rate.

Typical ESA TC rates are 2 and 4 kbps for deep space missions and 64 kbps for Earth Observation.

Low data rate is ok for commands, but beware…

- Software updates may take a long time
- Loading of long TC timeline also takes time
- OPS-SAT will be the first ESA mission to implement 256 kbps uplink

Note that TC frames can be short (8 to 1024 octets) and coding works better on long data sets. Why are TCs short?

- Enables short telecommands in critical situations (e.g. tumbling safe mode)
- Enables on line operations even at low data rate

Traditional Uplink Coding

BCH (Bose-Chaudhuri-Hocquenghem) is used – usually done mission control centre.

It comes in different flavours, SEC, TED and Soft. ESA uses SEC (Single Error Correction with 64 (63 + 1 spare) symbols for 56 bits of information i.e. 7 bytes information and one byte block code.

NASA use BCH in TED (Triple Error Detection). This means they do not need to employ CRCs on their telecommands while ESA does.

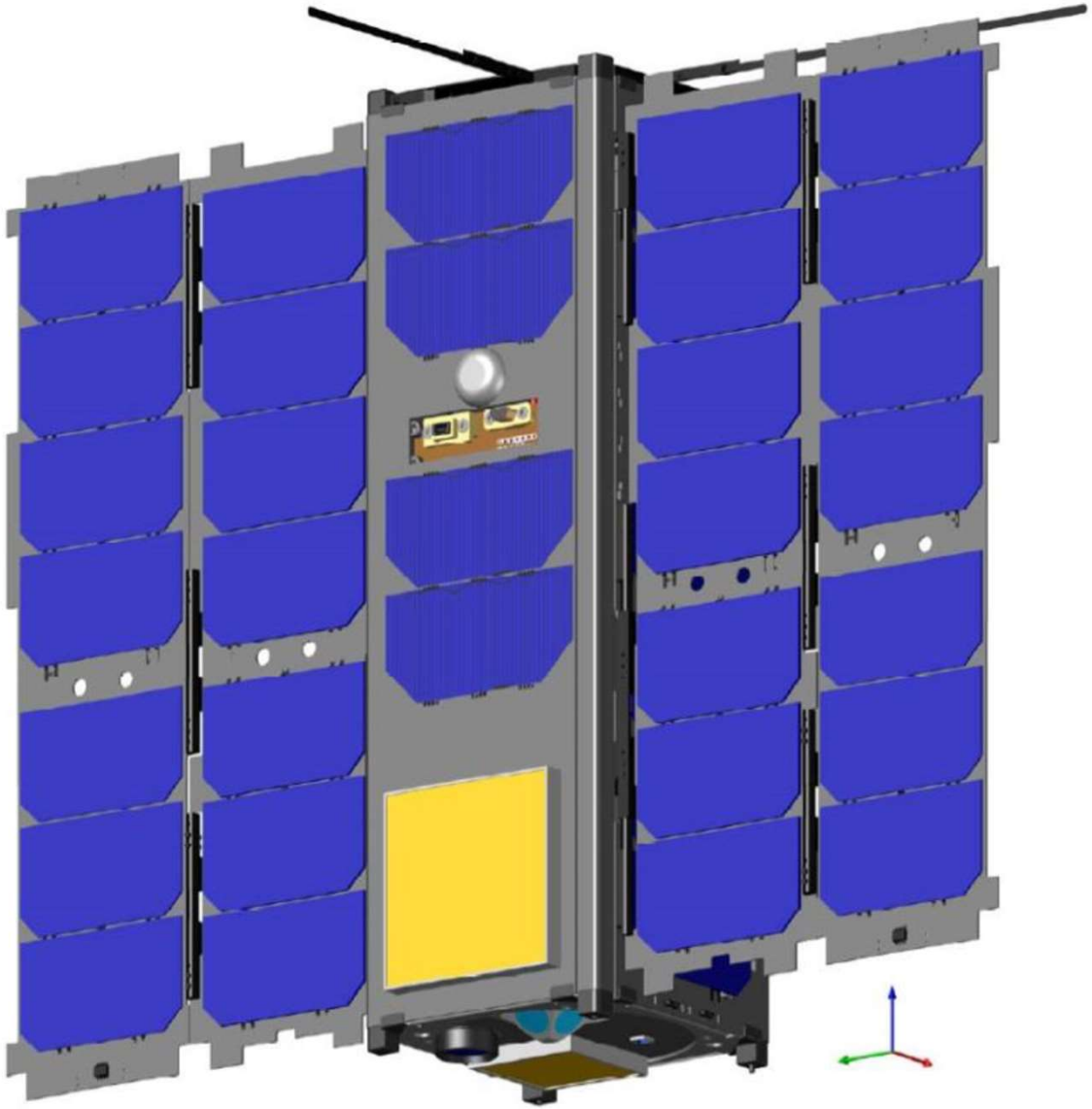
The coding rate on the ESA link is TC link is 0.875.

Bit Error Rate: BER = 10^{-5} (i.e. one error for 100 000 bits) for **Eb/No** of 9.6

The signal to noise ratio per bit is small compared to an un-coded signal (13.4 dB)

This is changing. CNES have successfully used convolution coding on some of their spacecraft uplinks. This allows better rates and some organisations are planning 1 Mbps uplinks. ESA is planning to use LDPC on the TC uplink for EUCLID.

Uplink Coding – OPS-SAT experience



Uplink Coding – OPS-SAT experience

Probability of detected error in a block

Probability of undetected error in a block

	E_b/N_o@		Req.	Block Length		Latency	Complexity	
Code	P_w=1e-3	P_u=1e-9	E_b/N_o	n	k	(bits)	Ops/bit	Memory
Uncoded	9.4	13.4	13.4	56	56	0	0	0
BCH, SEC	7.5	9.6	9.6	63	56	56	1	0.006
BCH, TED	9.9	7.7	9.9	63	56	56	2	0.069
BCH, soft	5.7	9 (est)	9 (est)	65	56	56		

OPS-SAT uses BCH, SEC

By implementing CRC checks on the TC following decoding we reduce the probability of an undetected error to below 1e-9. Therefore we are left with the Eb/No required for and so can exploit the extra coding gain i.e. 13.4dB - 7.5dB (instead of 9.6dB)

The lesson is that it pays to understand what the standards are trying to achieve rather than what they say sometimes..