Turbo Codes for Deep Space Communications: CCSDS 131.0-B-2 standard implementation

Final project for the Channel Coding course

Gianluca Marcon
gianluca.marcon.1@studenti.unipd.it
University of Padova
May 3, 2017

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

Standard specifications

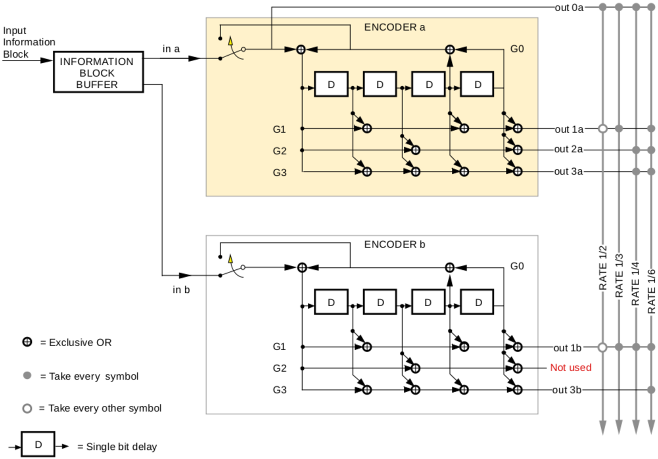
The standard specifies different input packet lengths k

- 1784
- 3568
- 7136
- 8920

...and different code rates R

- $1/2$
- $1/3$
- $1/4$
- $1/6$

Encoder structure

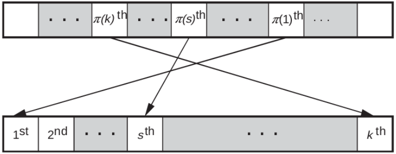


Convolutional codes defined through forward and backward connection vectors

Example: defining a code in C

```c
// define first code
int N_components = 2;
char *forward[N_components];
forward[0] = "10011";
forward[1] = "10101";

char *backward = "0011";

t_convcode code = convcode_initialize(forward, backward,
                                       N_components);
t_turbocode turbo = turbo_initialize(code, code,
                                      pi, info_length);
```

Interleaver



i-th bit of the interleaved packet is the $\pi(i)$-th bit of the original packet

Information block length	k_1	k_2
1784	8	223
3568	8	223×2
7136	8	223×4
8920	8	223×5

Building the interleaver

$$p = \begin{bmatrix} 31 & 37 & 43 & 47 & 53 & 59 & 61 & 67 \end{bmatrix}$$

for $s = 1$ to k **do**

$$m = (s-1) \mod 2$$
$$i = \text{floor}\left((s-1)/2k_2\right)$$
$$j = \text{floor}\left((s-1)/2\right) - ik_2$$
$$t = (19i+1) \mod (k_1/2)$$
$$q = t \mod 8 + 1$$
$$c = (p_q j + 21m) \mod k_2$$
$$\pi(s) = 2(t + ck_1/2 + 1) - m$$

end for

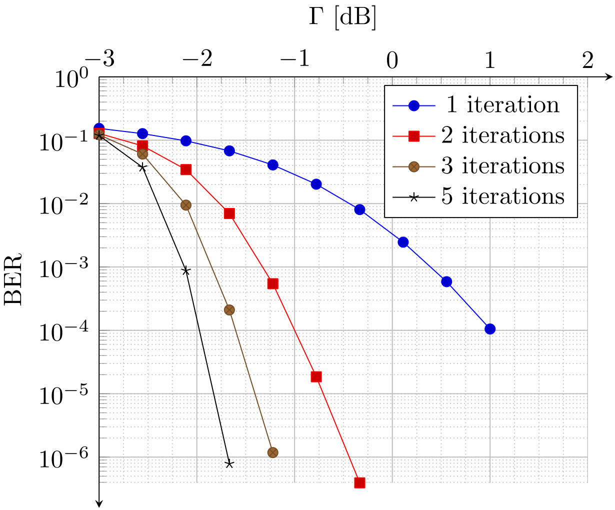
Decoding

- BCJR (in log domain) on upper and lower code
- scheduling as seen in class
- number of iterations is tuned accordingly
- puncturing is applied at reception for an easier implementation $\hat{r}[i] = r[i] \cdot p[i]$, $1 \leq i \leq (k+4)/R$

Example: defining a code in C

```c
int *decoded = NULL;
for (int i = 0; i < iterations; i++) {

    // run BCJR on upper code
    convcode_extrinsic(streams[0], lengths[0],
                       &messages, code.upper_code,
                       noise_variance, 0);
    // apply interleaver
    message_interleave(&messages, code);

    // run BCJR on lower code
    decoded = convcode_extrinsic(streams[1], lengths[1],
                       &messages, code.lower_code,
                       noise_variance,
                       i == (iterations - 1));
    // deinterleave
    message_deinterleave(&messages, code);
}
```

Effect of increasing number of iterations

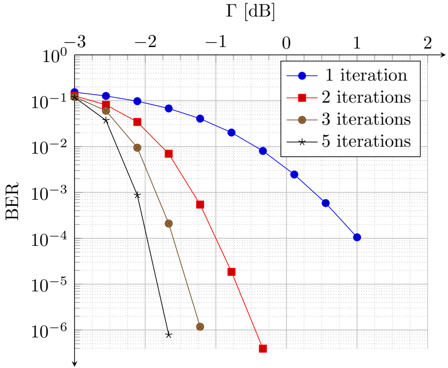


Effect of increasing number of iterations

```
gianluca@ursula  ~/deepspace-turbo/bin  master ●  ./deepspace_turbo --help
-y / --skip-confirm
        skip confirmation dialog after summarizing simulation parameters. Useful when automating simulations.

        -h / --help
        print this help dialog.

-o / --output FILENAME
        save results in a comma-separated format in FILENAME. If this argument is not used, the results will be saved in a file named with the current date and time.

-c / --packet-count INTEGER
        set the number of packets to encode/decode. INTEGER can be given in exponential notation i.e. 1e4 for 10000 packets.

-C / --cores INTEGER
        set the number of CPU cores to use.

-k / --multiplier INT
        set the input packet length through the following  formula: packet-length = 223 * 8 * multiplier

-l / --packet-length INTEGER
        set the number of information bits in a packet. Exponential notation can be used.

-m / --min-SNR FLOAT
        set the lower extreme of the SNR range to test.

-M / --max-SNR FLOAT
        set the upper extreme of the SNR range to test.

-n / --SNR-points INTEGER
        set the number of linearly spaced points inside the interval [min-SNR, max-SNR]

   -i / --iterations
        set the number of iterations for the turbo decoding algorithm.
```

Different packet sizes

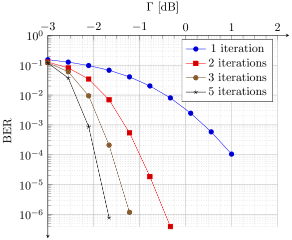
First type of modulator

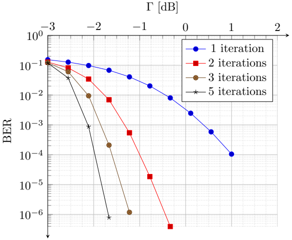


[Cho et al., 2011]

Large footprint required to obtain adequate modulation depth

Novel design: slot waveguides



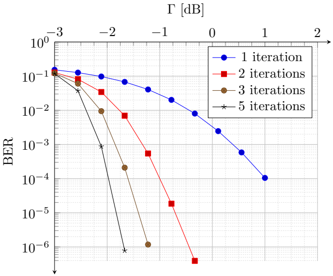


$\lambda = 1550$ nm, $n_{co} = 1.46$, $n_{cl} = 3.48$, $w_{co} = 101$ nm. [Xu et al., 2004]

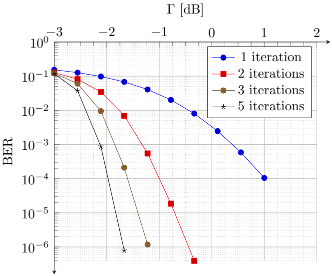
Absorbed power per unit area

$$P \propto \frac{1}{2}|\mathbf{E}| \cdot \frac{Im\{\varepsilon_{eff}\}}{|\varepsilon_{eff}|}$$

Wavelength and Voltage dependency

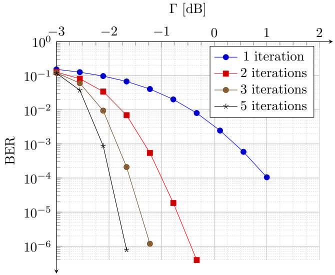


Bandwidth > 1.25 THz



$\Delta V = 1.32\,V$

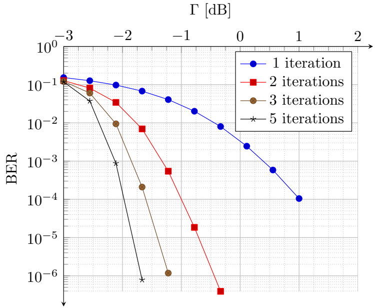
Plasmonic-graphene waveguide modulator



Cu is preferred (CMOS compatible), but has higher losses than Au, Ag.

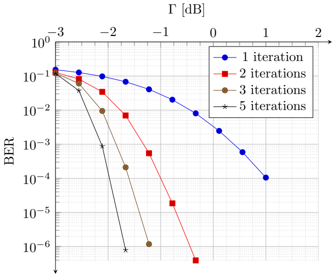
Plasmonic-graphene waveguide modulator

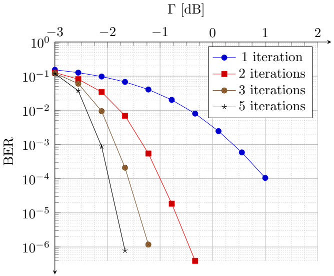
200 nm wide, silicone nitride 10 nm thick (both layers), length 120 nm.



Footprint $\sim 2 - 3\ \mu m^2$

Wavelength and Voltage dependency





Bandwidth > 1 THz

$\Delta V = 1.38\, V$

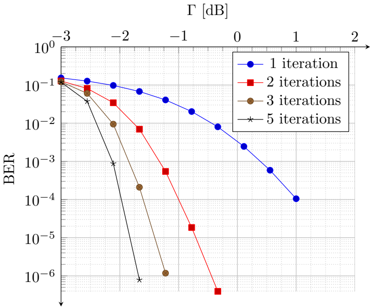
Energy consumption $\sim$ 0.12-0.13 pJ/bit

Did we meet any requirement?

- high bandwidth Yes
- energy efficiency Yes
- compatibility with on-chip electronic components Yes
- low insertion loss Yes
- small footprint Kinda
- high switching speed Potentially

Recent advances: graphene-on-silicon MZI

- Fix one arm in ”dielectric state” (low loss) of graphene
- Exploit changes in $Re\{\varepsilon_{eff}\}$ wrt gate voltage to induce phase change



[Phatak et al., 2016]

References I

Cho, S., Yoon, M. C., Kim, K. S., Kim, P., Kim, D., Ulin-avila, E., and Zentgraf, T. (2011).
A graphene-based broadband optical modulator.
Nat. (UK), 474(7349):64–6767.

Phatak, A., Cheng, Z., Qin, C., and Goda, K. (2016).
Design of electro-optic modulators based on graphene-on-silicon slot waveguides.
Optics Letters, 41(11):2501.

References II

Xu, Q., Almeida, V. R., Panepucci, R. R., and Lipson, M. (2004). **Experimental demonstration of guiding and confining light in nanometer-size low-refractive-index material.** *Optics letters*, 29(14):1626–1628.

Thank you!