Energy Consumption Analysis of Software Polar Decoders on Low Power Processors

Adrien Cassagne, Olivier Aumage, Camille Leroux,
Denis Barthou and Bertrand Le Gal

Exploring Soft ECC Decoding

Growing interest for Software Defined Radio (SDR)

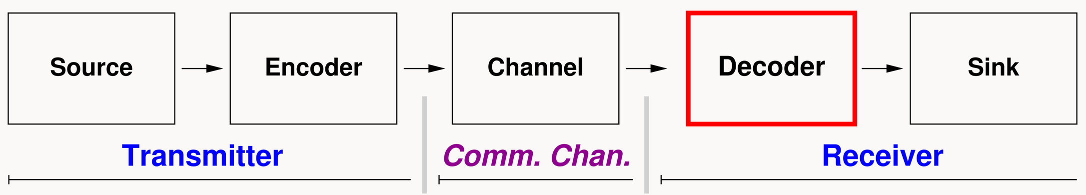

Figure 1: Simplified communication chain

- Leverage powerful, energy efficient procs (x86, ARM)
- Reduce dev. cost and time to market
- Validate and optimize new algorithms
- Enable Cloud computing-based architecture for Radio Access Networks (C-RAN)

Recent **Successive Cancellation** soft decoder for Polar codes [1] strongly benefit from modern CPUs capabilities and SIMD units, open the way to a wide optimization range.

Introducing AFF3CT, a software environment for exploring ECC decoders.

Decoding of Polar Codes

The **Successive Cancellation (SC)** decoding algorithm: a depth-first binary tree traversal algorithm based on 3 key functions:

$$\begin{cases} f(\lambda_a, \lambda_b) &= sign(\lambda_a.\lambda_b).\min(|\lambda_a|, |\lambda_b|) \\ g(\lambda_a, \lambda_b, s) &= (1-2s)\lambda_a + \lambda_b \\ h(s_a, s_b) &= (s_a \oplus s_b, s_b). \end{cases}$$

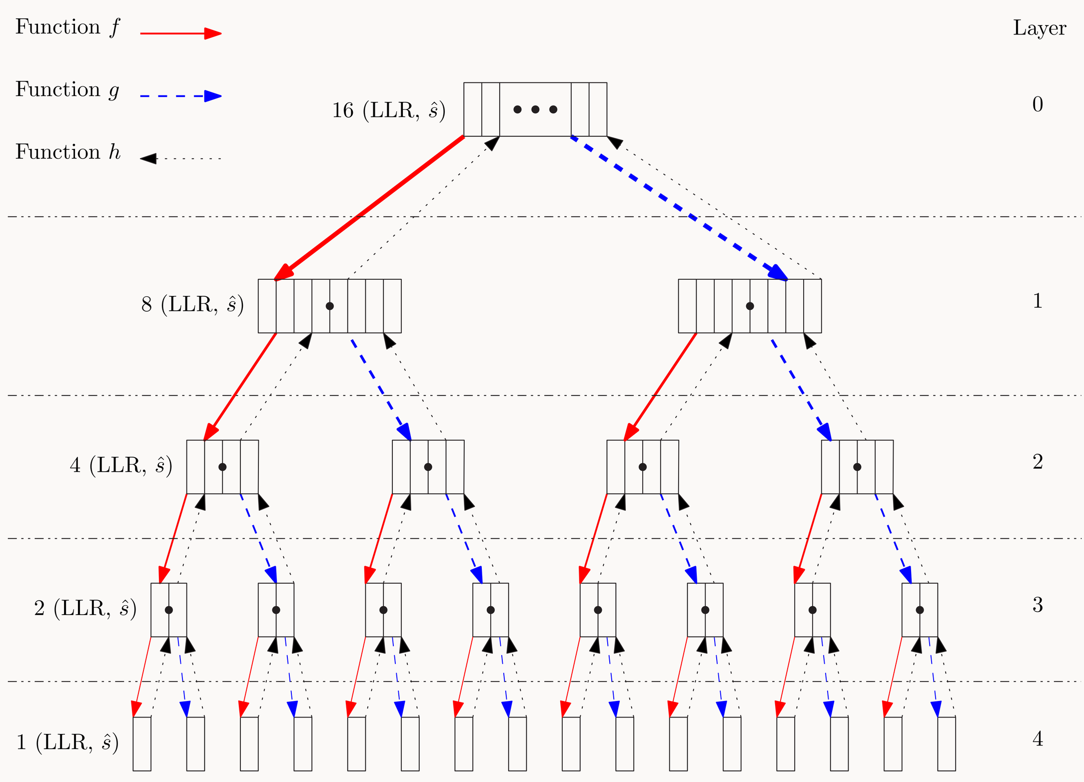

Figure 2: Full SC decoding tree ($N = 16$)

Conclusion

State of the art SC optimizations and performances

- Inter/intra-frame SIMD implementations
- Generated and dynamic decoders

Energy consumption analysis

- Software SC decoder = only **14 nJ per bit**, **65 Mbps** (ARM Cortex-A57 @ 1.1GHz, $N = 4096$, $R = 1/2$)
- Performance and energy consumption comparison on **big.LITTLE ARM32/64** and **Intel x86** processors

References

[1] E. Arikan. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE TIT*, 55(7):3051–3073, 2009.

[2] P. Giard, G. Sarkis, C. Thibeault, and W.J. Gross. Fast software polar decoders. In *Proc. of the IEEE ICASSP*, 2014.

[3] A. Cassagne, B. Le Gal, C. Leroux, O. Aumage, and D. Barthou. An efficient, portable and generic library for successive cancellation decoding of polar codes. In *Proc. of the Springer LCPC Work.*, 2015.

[4] B. Le Gal, C. Leroux, and C. Jego. Multi-Gb/s software decoding of polar codes. *IEEE TSP*, 63(2):349–359, 2015.

[5] G. Sarkis, P. Giard, C. Thibeault, and W.J. Gross. Autogenerating software polar decoders. In *Proc. of the IEEE GlobalSIP Conf.*, 2014.

Contribution of this Work

Fast and efficient implementations of the SC decoding algorithm on low power ARM processors

- Based on the Single Instruction Multiple Data (SIMD) CPU capability
  - **Intra-frame** [2, 3] (SIMD is used to compute many LLRs in a single frame): **low latency**, **moderate throughput**
  - **Inter-frame** [4, 3] (SIMD is used to process multiple frames in parallel): **high latency**, **high throughput**
- Specific SC algorithm optimizations: tree pruning or Fast-SSC [2] (rate 0, rate 1, single parity check and rep.)

Two different approaches are available:

- **Generated** [5, 3] (gen.): all the recursive calls are unrolled, specific decoder for a given SNR, faster decoders
- **Dynamic** [2, 4] (dyn.): the recursive calls are not unrolled, the decoder can adapt dynamically to various SNR

The first work to combine/compare all of these optimizations with energy considerations in mind

A Fast Forward Error Correction Tool (AFF3CT): Generic ECC Simulation Framework

AFF3CT: a software dedicated to simulations of digital communications with channel coding

`http://aff3ct.github.io`

- Support many codes: **Polar**, **Turbo**, **Convolutional**, **Repeat and Accumulate** and **LDPC** (coming soon)
- Very fast simulations, take advantage of today CPUs architecture (**hundreds of Mb/s on Intel Core i5/7**)
  - Written in `C++11` (**SystemC/TLM support**)
  - Monte-Carlo **multi-threaded** simulations
  - From **10 to 1000 faster than MATLAB** code
- **Portable**: run on Linux, Mac OS X and Windows
- **Open-source code** (under MIT license)

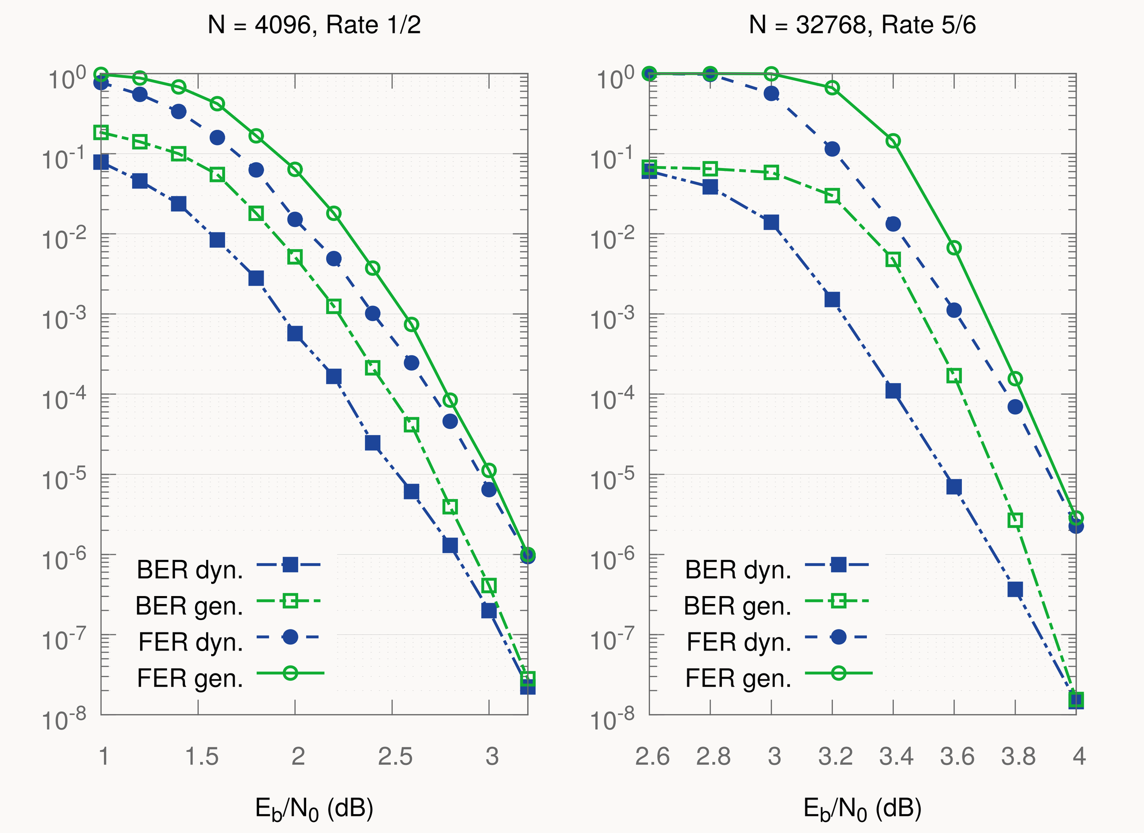

Figure 3: Simulated BER and FER for the Fast-SSC decoder

Experiments and Measurements

Cluster	Impl.	T_i (Mb/s)	l (μs)	E_b (nJ)	P (W)
A7-450MHz	seq.	3.1	655	37.8	0.117
	intra	13.0	158	9.5	0.123
	inter	21.8	1506	6.0	0.131
A53-450MHz	seq.	2.1	966	29.0	**0.062**
	intra	10.1	203	7.0	0.070
	inter	17.2	1902	**5.1**	0.088
A15-1.1GHz	seq.	7.5	274	122.0	0.913
	intra	35.2	58	28.2	0.991
	inter	62.8	522	17.4	1.093
A57-1.1GHz	seq.	9.2	222	78.9	0.730
	intra	39.2	52	21.1	0.826
	inter	65.1	503	14.2	0.923
i7-3.3GHz	seq.	36.3	56.5	235.4	8.532
	intra	221.8	**9.2**	40.5	9.017
	inter	**632.2**	51.8	15.8	9.997

Table 1: Characteristics for each cluster (T_i is the information throughput), for dyn. decoder. $N = 4096$, rate $R = 1/2$. The RAM consumption is not included in E_b and in P.

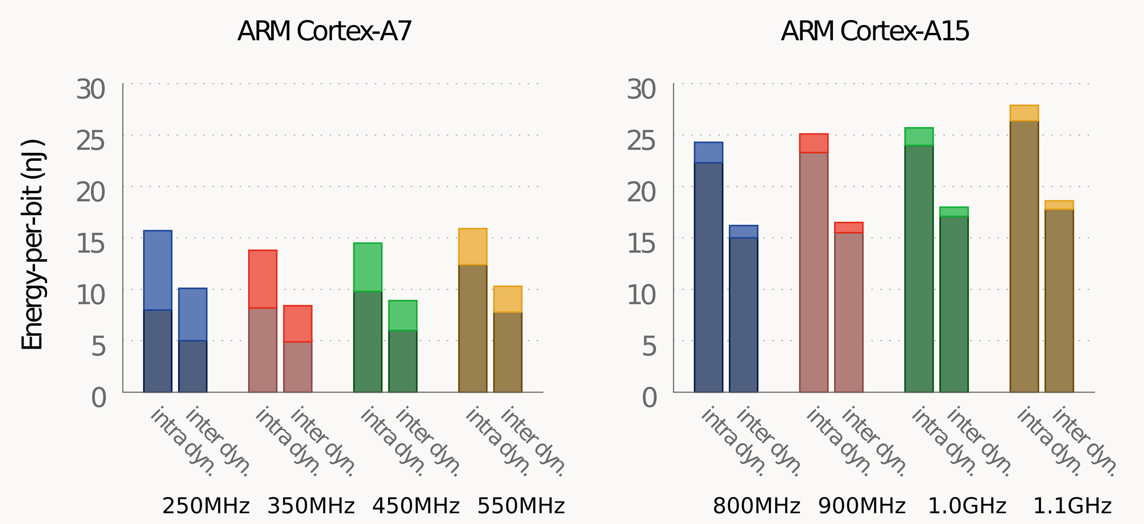

Figure 5: Variation of the *energy-per-bit* (E_b) depending on the cluster frequency (dynamic code, intra-, inter-frame). A7 performance is on the left and A15 on the right. $N = 4096$ and $R = 1/2$. Dark colors and light colors stand for CPU cluster and RAM energy consumption, resp.

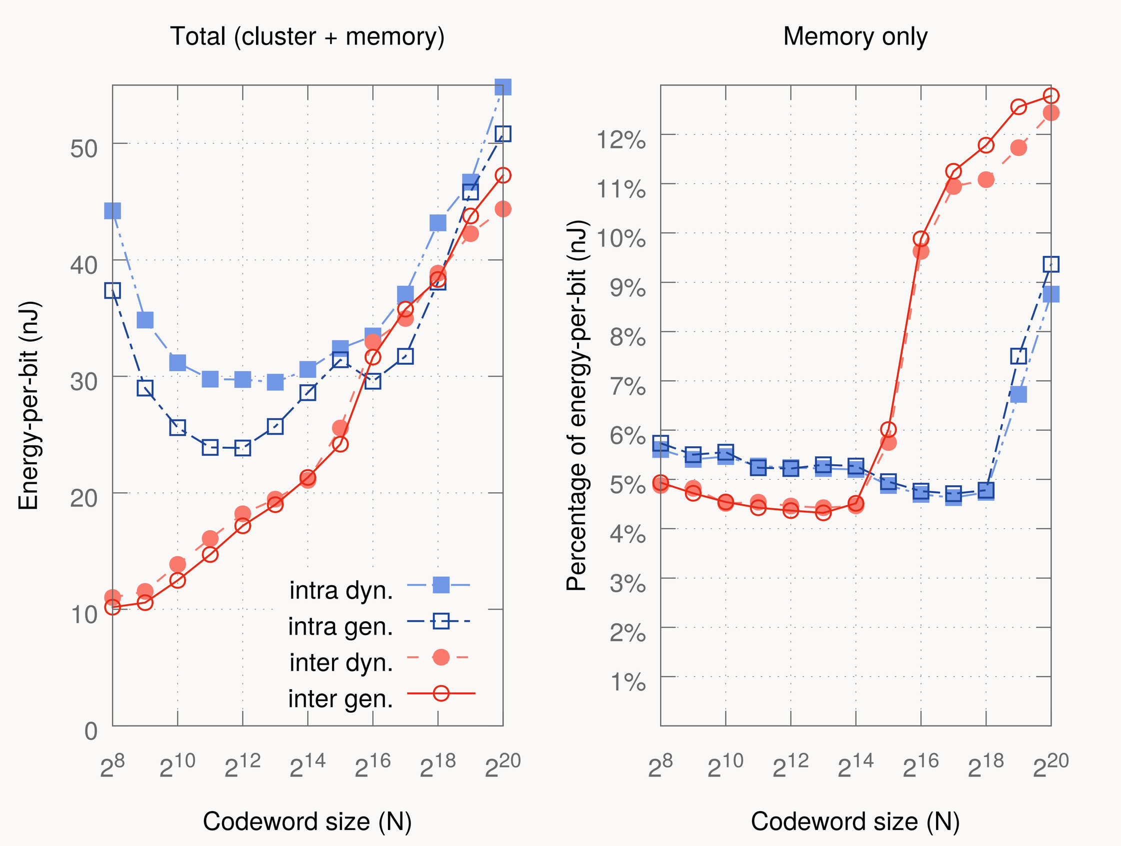

Figure 4: Variation of the *energy-per-bit* for different frame sizes and impl.: intra-/inter-frame, dyn. code on/off, on A15 @ 1.1GHz. Fixed rate $R = 1/2$.

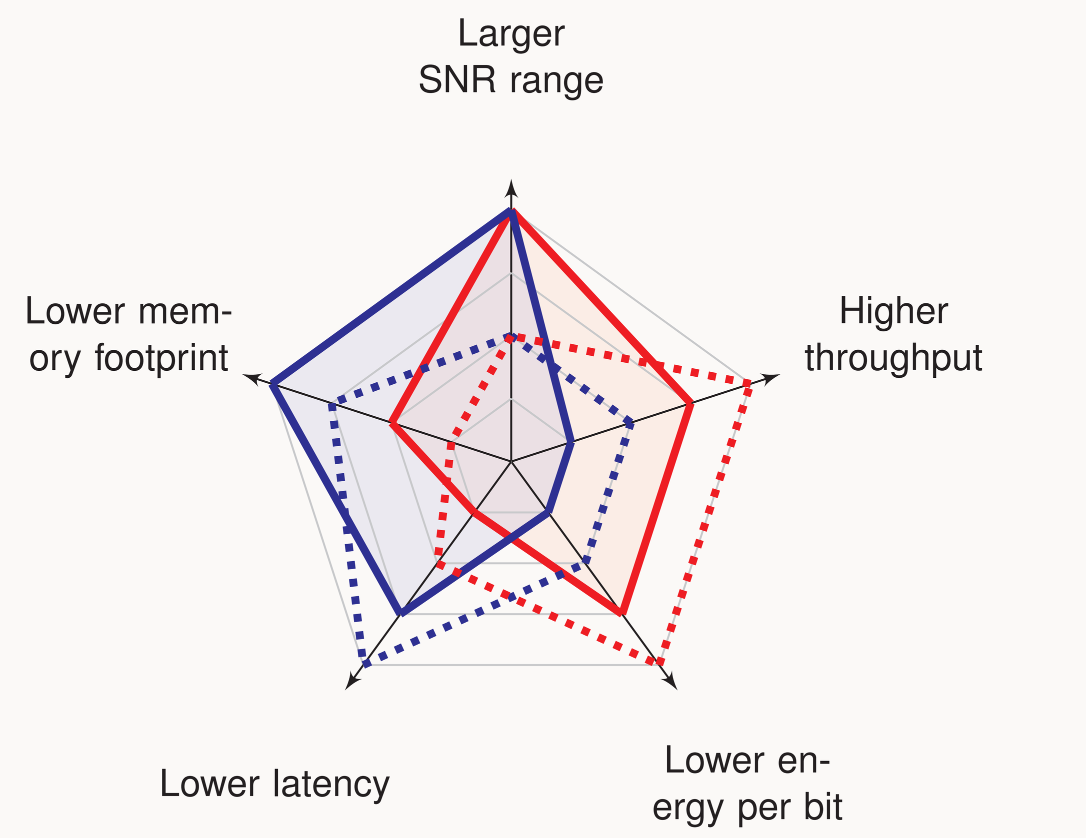

Figure 6: Ranking of the different approaches along 5 metrics. In red, inter-frame vectorization performance and in blue, intra-frame performance. Solid color is for the dynamic versions, dotted is for the generated versions. Each version is sorted along each of the 5 axes and the best version for one axe is placed further from the center.

Contact

Contact e-mail: `adrien.cassagne@inria.fr`

Acknowledgements

This study has been carried out with financial support from the French State, managed by the French National Research Agency (ANR) in the frame of the "Investments for the future" Programme IdEx Bordeaux - CPU (**ANR-10-IDEX-03-02**).