Fast simulation and prototyping with AFF3CT

Adrien Cassagne[⋆†] Olivier Hartmann[⋆] Mathieu Leonardon[⋆] Thibaud Tonnellier[⋆] Guillaume Delbergue[⋆]
Camille Leroux[⋆] Romain Tajan[⋆] Bertrand Le Gal[⋆] Christophe Jego[⋆] Olivier Aumage[†] Denis Barthou[†]
[⋆] IMS Lab, Bordeaux INP, France
[†] INRIA / LABRI, Univ. Bordeaux, INP, France

***Abstract*—This demonstration intends to present AFF3CT (A Fast Forward 3rror Correction Tool). The main objective of AFF3CT is to provide a portable, open source, fast and flexible software to the channel coding community in such a way that researchers can spend more time on channel coding / algorithmic problems instead of software development issues. It is also intended to facilitate the process of hardware verification and debug with the objective of fast prototyping.**

I. Simulation of a digital communication chain

Despite the wide variety of existing communication systems, all of them are based on a common abstract model that was proposed by the genius founder of information theory, Claude Shannon [1]. Figure 1 shows the synoptic of such a communication chain. In this structure, the channel encoder and decoder determine the achievable error rate of the system. Moreover, the channel decoder is a large contributor in the overall computational complexity of the system.

On the eve of the 5th generation of mobile communication systems, one of the challenges is to imagine systems able to transmit a huge amount of data in a very short amount of time at a very small energy cost in a wide variety of environments. In such a context, researchers work at refining some existing coding schemes (encoder + decoder) in such a way that the system has a low residual error rate and that the associated decoder is fast, flexible and has a low complexity.

The validation of a new coding scheme requires the estimation of the error rate performance. Unfortunately, most of the time, no simple mathematical model exists to predict the performance of a channel encoder/decoder. The only simple solution is to perform a Monte Carlo simulation of the whole communication chain: some data are pseudo-randomly generated, encoded, modulated, noised, decoded and the performance is estimated by measuring the Bit Error Rate (BER) and the Frame Error Rate (FER) at the receiver side. This apparently simple setup leads to three main problems.

Reproducibility: It is usually a tedious task to reproduce the results from the literature. This can be explained by the large amount of empirical parameters necessary to define one communication system and not all of them are reported in the publications. Moreover, it is rare that researchers actually share the source code of their simulator. As a consequence, a large amount of time is spent "reinventing the wheel" only to be able to compare to the state-of-the-art results.

Simulation time: In order to accurately estimate the FER/BER, one need to observe around 100 erroneous frames after decoding. This means that measuring a FER of say 10^{-7} requires to simulate in average the transmission of $100 \times 10^7 = 10^9$ frames. Assuming we are considering a frame of 1000 bits, the simulator needs to emulate the transmission of 10^{11} bits. Keeping in mind that the decoding algorithm can have a significant complexity, it happens that several weeks or months are required to estimate the FER/BER of a communication system.

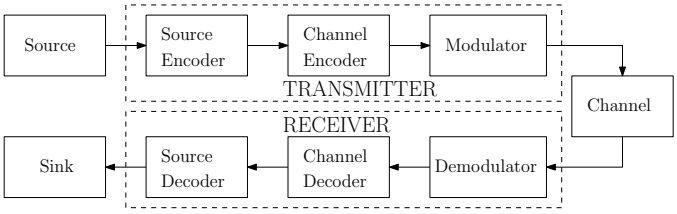

Fig. 1. Digital communication chain

Algorithmic heterogeneity: In the presented communication chain, there exists a large number of different channel codes. For each kind of code, several algorithms can be used for decoding. While it is quite simple to describe a unique coding scheme, it is more of a challenge to have a unified software description that supports all the coding schemes and the associated algorithms. This difficulty comes from the heterogeneity of data structures necessary to describe a channel code and the associated decoder : Turbo codes [2] use trellis, LDPC codes [3] are well defined on a factor graphs and polar codes [4] are efficiently decoded using binary trees.

II. Fast, portable, and flexible software simulation of communication systems

The reproducibility issue pushes towards an open source and portable software. As such, AFF3CT is available online [5] under an MIT license. This license actually allows anyone to download, modify or even sell the software. In terms of portability, the tool can run on several platforms (x86, ARM, Xeon Phi) and operating systems (Windows, MacOS and Linux). It can be compiled with various compilation tools (gcc, icc, clang and msvc).

In terms of simulation time, AFF3CT extensively uses software parallelization techniques : SIMD intrinsics, multi-threading and multi-node execution. On multi-node servers, the speed scales linearly with the number of available cores.

Regarding the flexibility, AFF3CT already supports most of the error correction codes included in the communication standards (turbo codes, LDPC, polar codes, BCH, convolutionnal codes...). Several modulation (PSK, QAM, CPM,...) and channel types (AWGN, Rayleigh) are also supported. Iterative demodulation/decoding is available for some coded modulation schemes. The sparse code multiple access (SCMA) technique is supported. Finally, AFF3CT can be used as a C++ library giving the designer complete freedom to design its own system using AFF3CT modules. In the perspective of hardware implementation, all modules can be described in SystemC. All these features can be used dynamically without having to recompile the source code.

III. Hardware verification and prototyping

A major challenge that arises when implementing a communication system in hardware is the verification step. In the case of error correction codes, it is a particularly difficult task due to the resilient nature of the algorithms. Actually, a channel decoder corrects errors. As a consequence, it is possible that an erroneous VHDL description seems to work because the algorithm itself corrects errors. In order to verify a hardware decoder, one should generate a large number of test patterns and perform a bit true comparison. AFF3CT allows to verify a design with the concept of "hardware in the loop". This allows to easily perform the co-simulation of a communication system: some modules can be described in software while some others are implemented on FPGA devices. This provides a simple way to verify and debug a hardware design. This protocol can be used at any stage of the communication chain to verify one or several modules.

IV. AFF3CT demonstration

The objective of the demonstration is to highlight some of the key features of AFF3CT such as the simulation speed, the flexibility and the portability.

A. High speed software simulation on an x86 and ARM processor

In the first setup, AFF3CT runs on a simple laptop. The basic functionalities of the tool are explored : changing the code type (turbo, LDPC, polar, ...), the decoder algorithm (BP, SC, SCL, ...), the channel (AWGN, Rayleigh), ... For each configuration, the error correction performance appears dynamically as the simulation runs. The activation of different run-time options shows how to speedup the simulation.

In order to show the portability of the tool, in the second setup, AFF3CT runs on an ARM processor. The selected board is XXX. The simulation speed is naturally lower than an x86 implementation but considering the power consumption, the consumed energy is lower. This implementation is an efficient solution for implementing a transmitter/receiver in an energy-constrained software radio context.

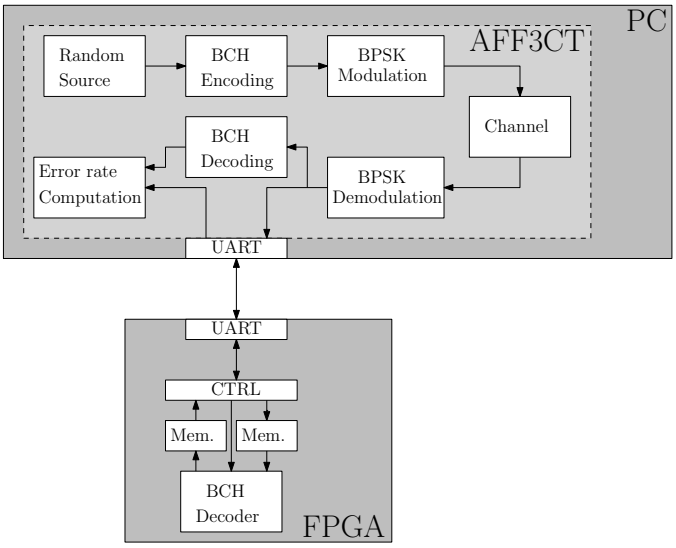

Fig. 2. Co-simulation of a BCH decoder

B. Co-simulation and prototyping of a BCH decoder

The objective of this last setup is to show how AFF3CT can be used to perform hardware verification and prototyping of a communication chain. In this setup, we consider a simple coding scheme including a BCH encoder and decoder. At first, a pure software simulation is launched. The error correction performance is plotted dynamically. Then, a Nexys-4 development board, including a Xilinx Artix-7 FPGA, is connected to the laptop. A hardware BCH decoder in the form of a bitstream is then downloaded to the FPGA. Once configured, the simulation is launched from the PC. AFF3CT simulates the transmission of a first frame. The noisy frame is then sent to the FPGA using the UART protocol. The hardware BCH decoder processes the frame and sends it back to the PC. AFF3CT can then proceed and perform the rest of the processing for this frame. Once the number of residual errors is updated, AFF3CT starts with a new frame, and so on. The decoding performance appears to be equivalent to the performance of the pure software simulation which shows that the hardware BCH decoder is correctly implemented.

Acknowledgment

This work was funded by ANR NAND (ANR-15-CE25-0006-01).

References

[1] C. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, 1948.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *IEEE ICC'93 Geneva.*, vol. 2. IEEE, 1993, pp. 1064–1070.

[3] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[4] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, 2009.

[5] AFF3CT : A Fast Forward 3rror Correction Tool. http://aff3ct.github.io.