Hardware Implementation and FPGA Prototyping of an Expectation Propagation-based Receiver

Ian Fischer Schilling, Serdar Sahin, Camille Leroux, Antonio Maria Cipriano, Christophe Jego

Hardware Implementation and FPGA Prototyping of an Expectation Propagation-based Receiver

Ian Fischer Schilling*, Serdar Şahin†, Camille Leroux*, Antonio Maria Cipriano†, Christophe Jégo*
*University of Bordeaux, Bordeaux INP
IMS Lab, UMR CNRS 5218, France
firstname.last-name@ims-bordeaux.fr
† Thales
Gennevilliers, France
firstname.last-name@thalesgroup.com

***Abstract*—This paper presents a hardware implementation and FPGA prototyping of a flexible pipelined Expectation Propagation (EP)-based receiver that can handle QPSK, 8-PSK and 16-QAM constellations. The receiver implements a Frequency Domain (FD) Self-Iterated Linear Equalizer (SILE) based on EP to approximate the true posterior distribution of the transmitted symbols by a simpler distribution. Analytical approximations for the EP feedback generation process and the three constellations are applied to reduce the hardware complexity of the soft mapper/demapper architectures. The implementation results show a significant reduction in clock cycles due to the pipeline and an efficient usage of FPGA resources, underscoring the effectiveness of the proposed architecture.**

***Index Terms*—Expectation Propagation, Frequency Domain Self-Iterated Linear Equalizer, Hardware Implementation, Architecture Design, FPGA prototyping, Pipelined Architecture**

I. Introduction

In digital communication systems, achieving minimal error rates in data detection and/or decoding requires the resolution of a Maximum A Posteriori (MAP) or Maximum Likelihood (ML) problem [1]. However, the computational complexity of resolving such criteria is often prohibitive, particularly in real-world frequency selective channels. In this context, the number of computations increases exponentially with factors such as data length, modulation order and channel memory. As a result, practical receiver design often involves the application of simplifying hypotheses and approximations. One promising approach in the context of Frequency Domain (FD) Linear Equalization (LE) is equalizers designed with Expectation Propagation (EP). Indeed, they have demonstrated an appealing trade-off between performance and computational complexity [2].

The implementation of a simplified EP receiver for multiple antenna receivers has been reported in [3], and a low complexity EP detector for sparse code multiple access was also proposed in [4]. Furthermore, simplified EP-based FD equalization is studied in [5]. However, these studies only provided a computational complexity assessment.

A comprehensive study and implementation of an EP-based receiver for communication over frequency-selective channels with standard Phase-Shift Keying (PSK) or Quadrature Amplitude Modulation (QAM) constellations was given in [6]. This receiver implements an FD Self-Iterated Linear Equalizer (SILE), where EP is applied to approximate the true posterior distribution of the transmitted symbols by a simpler distribution that can be easily manipulated.

Building on previous work [6], a hardware implementation of a flexible pipelined EP-based receiver is detailed in this paper. To the best of our knowledge, this is the first EP-based FD-SILE architecture that can accommodate three different constellations.

This work has been funded by the French National Research Agency under grant number ANR-20-CE25-0008-01 (EVASION Project: https://anr-evasion.ims-bordeaux.fr/).

II. Simplified EP-based FD-SILE

A. General Structure

Expectation Propagation (EP) is a powerful technique exploited in statistical inference to approximate complex probability distributions by simpler distributions from the exponential family through moment matching. The Frequency Domain (FD) Self-Iterated Linear Equalizer (SILE) algorithm derived in [2] is based on EP to compute extrinsic soft decision feedback.

The functional structure of the iterative receiver, which is based on the analytical simplifications described in [6], is depicted in Fig. 1. The received signal undergoes a transformation to the frequency domain via an FFT function. Subsequently, a linear Minimum Mean Square Error (MMSE) filter with interference cancellation is employed for equalization. In addition to the channel frequency response and noise statistics, the filter computation requires the statistics of the soft decision feedback [2].

The receiver executes S self-iterations, that pass through the equalizer, the soft demapper, and the EP-based soft mapper. After the filtering stage, the equalized symbols in the time domain $x_k^{e(s)}$ are processed by the soft demapper, along with

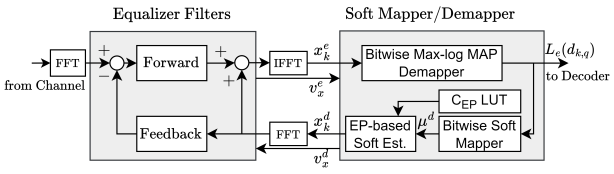
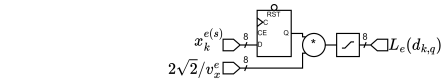
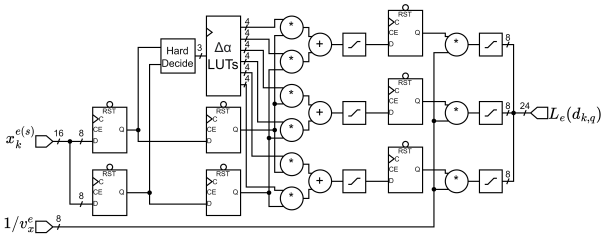


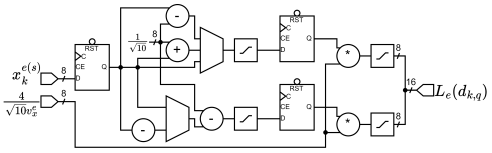
Fig. 1: Simplified EP-based FD-SILE functional structure.



(a) QPSK.



(b) 8-PSK.



(c) 16-QAM.

Fig. 2: Architectures of Bitwise Max-log MAP Demapper.

an estimate of the residual post- equalization noise and interference variance $v_x^{e(s)}$. The bitwise max-log MAP demapper then analytically estimates the Log-Likelihood Ratios (LLRs) $L_e(d_{k,q})$ based on the constellation under consideration [6].

During the last self-iteration, the LLRs are sent to the FEC decoder. However, during the iterative process, the LLRs are input to the bitwise soft mapper, where the soft symbol estimate μ^d is analytically computed according to the constellation. The EP-based soft estimates are then calculated using the equalized symbols in the time domain $x_k^{e(s)}$, the estimate of the residual post- equalization noise and interference variance $v_x^{e(s)}$, the soft symbol estimate μ^d, and the auxiliary quantity $C_{EP}(v_x^e) = \tilde{\gamma}_x^d/(v_x^e - \tilde{\gamma}_x^d)$, which is tabulated in terms of $v_x^{e(s)}$, where $\tilde{\gamma}_x^d$ is the asymptotic a posteriori mean square error (MSE) [5] [6].

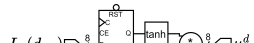
The focus of the work lies on the blocks within the Soft Mapper/Demapper.

B. Bitwise Max-log MAP Demapper

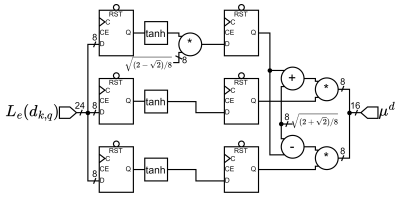
The proposed bitwise max-log MAP demapper, as shown in Fig. 2, processes the equalized symbols and generates the soft bits. The architectures enable the processing of the analytical expressions [6] of the QPSK (Fig. 2a), 8-PSK (Fig. 2b) and 16-QAM (Fig. 2c) constellations.

For the 8-PSK constellation, the demapper starts by retrieving the precomputed values of $\Delta_{\alpha^*,q}$ from a Look-Up Table (LUT) according to the max-log MAP criterion [6]. Then it proceeds to multiply the equalized symbols by the LUT values and add the real and imaginary results, followed by a saturation block.

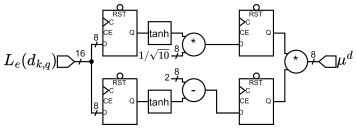
In the case of the 16-QAM constellation, one of the LLRs is calculated using the absolute value of the equalized symbols, while the second LLR is determined based on a conditional statement that depends on the modulus of the equalized symbols. Both results are then passed through a saturation block.



(a) QPSK.



(b) 8-PSK.



(c) 16-QAM.

Fig. 3: Architectures of Bitwise Soft Mapper.

At the end of each LLR calculation, a multiplication operation is performed between the results and a tabulated constant divided by sigma, in order to avoid division operations. In the QPSK constellation, for example, this value is given by $2\sqrt{2}/v_x^e$. This is followed by a saturation block.

The execution time of the demapper varies depending on the constellation: QPSK requires one clock cycle, 8-PSK requires three clock cycles and 16-QAM requires two clock cycles. The registers were added to remove the critical paths of the architecture and thus operate at a frequency of 100 MHz.

C. Bitwise Soft Mapper

The proposed bitwise soft mapper, as shown in Fig. 3, operates on the LLRs and generates soft symbol estimates, denoted as μ^d. The architectures enable the processing of the analytical expressions described in [6] of the QPSK (Fig. 3a), 8-PSK (Fig. 3b) and 16-QAM (Fig. 3c) constellations.

The operation begins by computing the hyperbolic tangent of the LLRs thanks to a piecewise linear approximation where all the slope and bias coefficients are powers of two. This enables the evaluation of the probability of each bit. The first probability is then multiplied by specific constants: $1/\sqrt{2}$ for QPSK, $\sqrt{(2-\sqrt{2})/8}$ for 8-PSK, and $1/\sqrt{10}$ for 16-QAM.

For the 8-PSK constellation, the product of the first probability and $\sqrt{(2-\sqrt{2})/8}$ is added to $\sqrt{(2+\sqrt{2})/8}$ and subtracted from it. Then the results are multiplied by the second and third probabilities, respectively. As for the 16-QAM constellation, the second probability is subtracted from 2 and then multiplied by the product of the first probability and $1/\sqrt{10}$.

The execution time of the bitwise soft mapper varies depending on the constellation: QPSK requires one clock cycle, while 8-PSK and 16-QAM require two clock cycles. The

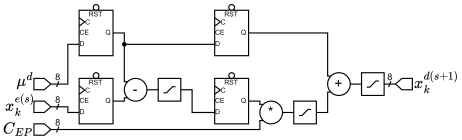


Fig. 4: Architecture of EP-based Soft Estimates.

registers were allocated once to operate at a frequency of 100 MHz.

D. EP-based Soft Estimates

The proposed *EP-based Soft Estimates* block, as shown in Fig. 4, employs the equalized symbols in the time domain $x_k^{e(s)}$, the soft symbol estimate μ^d, and the auxiliary quantity $C_{EP}(v_x^e)$. The operation begins with the subtraction of the equalized symbols $x_k^{e(s)}$ from the soft symbol estimates μ^d and the subsequent saturation of the result. This is followed by a multiplication of the result by the auxiliary quantity C_{EP} and another saturation. Finally, the result of the multiplication is added to the soft symbol estimates μ^d and the final result is saturated. This process is consistent across all constellations and is designed to operate at a frequency of 100 MHz, taking two clock cycles to execute.

III. A Flexible Pipelined Architecture

The flexible pipelined architecture, as depicted in Fig. 5, is designed to support any of the three constellations mentioned: QPSK, 8-PSK, and 16-QAM. The receiver can dynamically change the mapping in each frame, thanks to the multiplexing logic that allows the selection of the appropriate data path.

In order to easily prototype the receiver on a Multi Processor System on Chip (MPSoC), a 32-bit AXI-stream [7] interface was employed. This interface consists of an AXI-stream master and an AXI-stream slave, both of which are equipped with the signals data, valid, last, and ready. A Finite State Machine (FSM) controls three states: Reset, Receive, and Send. This requires the valid, last, and ready signals of both the master and the slave for the state transitions.

The *Control Data* block receives the initial message, which conveys whether there is a self-iteration, the constellation identifier, and the 8-bit value of the estimated variance, $v_x^{e(s)}$. The variance value is then processed by a ROM that contains the auxiliary quantity C_{EP}. The C_{EP} is used to calculate the extrinsic soft feedback variance $v_x^{d(s+1)}$ in the *Extrinsic Variance* block and the extrinsic soft feedback estimates $x^{d(s+1)}$ in the *EP-based Soft Estimates* block (Fig. 4). The variance is also processed by a ROM containing the inverse variance values for the demapper, as shown in Fig. 2.

All messages are expressed in 32 bits. The messages following the first one contain two complex symbols each. These two complex values are processed in parallel and in a pipeline. First they pass through the demapper of their respective constellation. If there is no self-iteration, the LLRs are moved to the master's data. However, if a self-iteration is required, the LLRs are transferred to the mapper of their respective constellation and then to the *EP-based Soft Estimation* block, and finally shifted out to the master's data.

The task scheduling for a QPSK frame with self-iteration is shown in Fig. 6. The process comprises two main phases: handling the control data and processing the symbol data.

The control data σ, being the first one, is initially received in the *s_axis_data*. It then moves to the *Control Data* block in the next clock cycle. Subsequently, σ navigates through the C_{EP} ROM and the *Extrinsic Variance* block, as well as the ROM of each constellation. By the fourth clock cycle, the processed control data is available for transmission. However, it is placed in *m_axis_data* only in the clock cycle before the first processed symbol data, which enables better utilization if there are multiple frames.

The symbol data is processed after the control data. Starting from the second clock cycle, *s_axis_data* has the first symbol data, denoted as S_1 and S_2. As the clock cycles progress, it moves through the demapper, the mapper, and is then processed by the EP-based soft estimation. By the seventh clock cycle, it is ready for transmission and is placed in *m_axis_data*.

This sequence is repeated for all symbols up to S_{128}, with each symbol pair following the path of its predecessor with a delay of one clock cycle. As each data contains 2 symbols, this results in a total number of 64 data values. The task scheduling for the other constellations is similar. 8-PSK requires two additional clock cycles for the demapper and one for the mapper, while 16-QAM requires one additional clock cycle for both the demapper and the mapper.

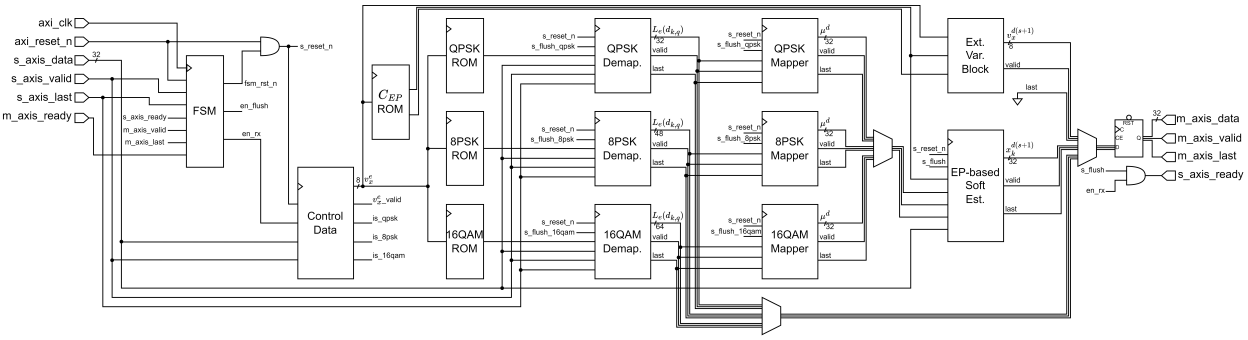


Fig. 5: Architecture of Flexible Pipelined Soft Mapper/Demapper.

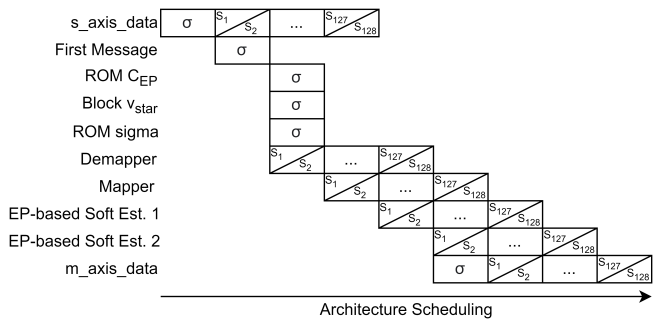


Fig. 6: Task Scheduling of Proposed Architecture for QPSK Constellation with Self-Iteration.

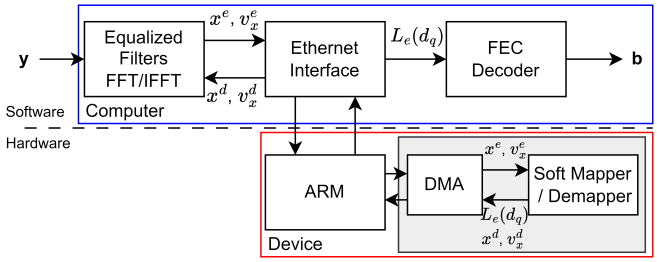


Fig. 7: Simplified EP-based FD-SILE FPGA and HIL Block Diagram.

IV. FPGA Implementation and Prototyping

A. Experimental Setup

The experimental setup employs a Hardware-in-the-Loop (HIL) configuration, involving a computer and a PYNQ Z2 board [8]. The computer runs py-AFF3CT, a Python wrapper for the Forward Error Correction Toolbox AFF3CT [9]. The PYNQ Z2 board is a prototyping board based on the Xilinx Zynq System on Chip (SoC) with an ARM processor and an FPGA ZYNQ XC7Z020-1CLG400C. As shown in Fig. 7, the board is connected to the computer via an Ethernet interface. The board's ARM processor forwards the data to the FPGA device via a Direct Memory Access (DMA).

Within the FPGA device, the Soft Mapper / Demapper block encapsulates the flexible pipelined architecture depicted in Fig. 5. The remaining parts of the receiver are executed thanks to AFF3CT on the associated computer. These software blocks encompass filtering, equalization, FFT and IFFT operations, rate dematching and Forward Error Correction (FEC) decoding. This HIL setup is useful to verify that the error rate performance matches the software reference model described with the AFF3CT library.

B. FPGA Implementation Results

Table I summarizes the allocated FPGA resources for each of the three constellations considered in this article: QPSK, 8-PSK, and 16-QAM. The values are given for the implementation of only the Soft Mapper / Demapper without the HIL configuration. The HIL configuration employs about 3200 LUTs, 5200 Flip-Flops and 2 BRAMs for each implementation. The implemented architectures assign a data width of 32 bits for the DMA connection. It enables the exchange of two symbols with 8-bit real and imaginary parts.

XC7Z020-1CLG400C	Number	QPSK		8-PSK		16-QAM		**Flexible Architecture**
		Orig.	Pipe.	Orig.	Pipe.	Orig.	Pipe.	
LUTs	53200	1035	1070	1805	1857	1448	1714	**3834**
Flip-Flops	106400	198	281	346	566	340	562	**1059**
BRAMs	0.5	0.5	0.5	0.5	0.5	0.5	0.5	**1.0**
Cycles	-	390	70	646	73	582	72	**70-73**

Frequency = 100 MHz

Table I: FPGA Resource Usage and Number of Clock Cycles for the Soft Mapper / Demapper.

In addition to the first implementations given in [6], pipelined versions of each constellation have been implemented, as well as a global flexible pipelined architecture that addresses the three mappings. While the pipelined versions consume slightly more resources, they significantly reduce the number of clock cycles required for the computations. The global flexible implementation, on the other hand, allocates fewer resources than the sum of the three individual implementations. Moreover, it offers the advantage that it can switch the constellation for each received message. These enhancements demonstrate the versatility and efficiency of the proposed architectures.

V. Conclusion

This paper details a hardware implementation of a flexible pipelined EP-based receiver. To the best of our knowledge, this is the first architecture that enables to consider three different constellations. Thanks to the hardware-in-the-loop approach, this architecture has been prototyped onto an MPSOC FPGA to estimate its BER performance. In addition to the initial block implementations, pipelined versions for each constellation have been proposed. These versions, albeit requiring slightly more resources, have significantly reduced the number of clock cycles.

References

[1] L. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", IEEE Trans. on Information Theory, Mar. 1974.

[2] S. Şahin, A. M. Cipriano, C. Poulliat, M.-L. Boucheret, "A framework for iterative frequency domain EP-based receiver design", IEEE Trans. on Communications, Dec. 2018.

[3] D. Auras, R. Leupers and G. Ascheid, "A Novel Class of Linear MIMO Detectors with Boosted Communications Performance: Algorithm and VLSI Architecture," 2014 IEEE Computer Society Annual Symposium on VLSI, 2014.

[4] J. Xiao, J. Hu, and K. Han, "Low complexity expectation propagation detection for SCMA using approximate computing", 2019 IEEE Global Commun. Conf. (GLOBECOM), 2019.

[5] A. M. Cipriano, S. Şahin, C. Poulliat, "Practical Frequency-Domain Decision Feedback Equalization Based on Expectation Propagation", IEEE Communications Letters, Oct. 2023.

[6] I. F. Schilling et al., "Hardware Implementation of Soft Mapper/Demappers in Iterative EP-based Receivers", arXiv Preprint, 2024, arXiv:2406.07934

[7] ARM, 2021. "AMBA AXI-Stream Protocol Specification". Available at: developer.arm.com/documentation/ihi0051/latest/

[8] AMD, 2024. AUP PYNQ-Z2. Available at: amd.com/en/corporate/university-program/aup-boards/pynq-z2.html

[9] A. Cassagne et al., "AFF3CT: A Fast Forward Error Correction Toolbox!," SoftwareX, 2019.