A common operator for FFT and FEC decoding

Malek Naoues, Dominique Noguet, Laurent Alaus, Yves Louët

A Common Operator for FFT and FEC Decoding

Malek Naoues, Dominique Noguet, Laurent Alaus
CEA-LETI, Minatec
38054 Grenoble Cedex 9, France
{malek.naoues;dominique.noguet;laurent.alaus}@cea.fr

Yves Louët
SUPELEC Campus de Rennes
35511 Cesson-Sevigné, France
yves.louet@supelec.fr

Abstract—**In the Software Radio context, the parametrization is becoming an important topic especially when it comes to multi-standard designs. This paper capitalizes on the Common Operator technique to present new common structures for the FFT and FEC decoding algorithms. A key benefit of exhibiting common operators is the regular architecture it brings when implemented in a Common Operator Bank (COB). This regularity makes the architecture open to future function mapping and adapted to accommodated silicon technology variability through dependable design.**

Keywords-Parametrization; Common Operator; FFT; Viterbi; RS decoding; Software Radio; Flexible Radio

I. INTRODUCTION

Over the past few years, a proliferation of communication standards has substantially increased the complexity of radio design. In typical designs, the communication standards are implemented separately using dedicated instantiations which are difficult to upgrade for the support of new features. In the present days, the concept of Software Radio (SWR), introduced by J. Mitola in [1], emerged from military research to become a cornerstone of modern communication system design. The SWR technique becomes the way to design flexible and reconfigurable architectures capable of supporting different transmission standards in a single platform. Although there is a common agreement on the SWR aim and benefit, the way of implementing SWR, also known as Software Defined Radio (SDR) varies, considering various tradeoffs requested by actual design (cost, flexibility, complexity, power consumption, speed, etc.), and current silicon technology.

A digital communication baseband chain, when supporting different standards, uses typical signal processing operations such as modulation, channel coding, equalization, etc. These functions can be identified and then explored to take advantage from the similarities among common tasks in order to enhance power efficiency and area occupation [4]. In this context, parameterization technique has been introduced in [2] and [3]. It consists in identifying the common aspects among the targeted modes and standards in order to define a generic operation capable of handling the required tasks. This generic operation can switch from a configuration to another by a simple change of its parameters.

In this paper, we exploit a parameterization approach proposed in [4], called the common operator technique that can be considered to build a generic terminal capable of supporting a large range of communication standards. The main principle of the common operator technique was to identify common elements based on smaller structures that could be heavily reused across functions. This technique aims at designing a scalable transceiver based on medium granularity operators, larger than basic logic cells and smaller than Velcro Method or common function [4]. Similarly to flip flop or logic gate, a common operator is used regardless of the function executed by. From this point, the common operator technique claims to be less standard dependent than classical approach [5] where the entire specific building block required by a standard are implemented and executed when needed. It is expected that the reduction of the exploration space to telecommunication baseband functions will help exposing medium-grain common operators. The resulted implementation is expected to be more flexible and scalable to a wide range of standards. Such a regular structure is also well adapted to cope with silicon technology process variability. Indeed, as CMOS technology shrinks, the performance of the operator instances may vary across space (on the silicon wafer) and time [6]. Dealing with regular building blocks helps map the most demanding algorithms onto the best performing cells, enabling the design to be dependable or even self-healing. Many previous works focused on defining [7,12,14,15] implementing and managing [8] the Common Operators (CO). In this paper we investigate the commonalities of the FFT and FEC decoding operator. The core of the paper focuses on a new operator that exploits similarities between FFT butterflies and trellis decoding structures used in the Viterbi algorithm. These similarities are exploited to suggest a CO for the FFT and the Viterbi decoder. CO for FFT and FEC was already studied in [13,16] with a focus on Reed-Solomon (RS) decoder based on a FFT operator over GF(2^m). This work is recapped herein to highlight how it can be considered along with the FFT/Viterbi CO to build a more general library framework for FFT/FEC functions.

Then, the paper is organized as follows. The second section presents the Common Operator technique. In the third section we briefly recap the work of [13,16]. Then, in section four we focus on the new FFT/Viterbi CO, exposing the similarities and their exploitation to build a new CO. The fifth section proposes a set of two common structures for FFT and FEC decoding algorithms; finally the results and the performances of these common operators are discussed in the last section.

II. THE COMMON OPERATOR TECHNIQUE

The conventional approach to implement a multi-standard radio device is to instantiate multiple transceiver chains each dedicated to an individual mode or standard (Fig.1). With this approach most of the hardware needs to be redesigned whenever an additional standard is to be considered. This conventional approach called "Velcro" does not exploit any

common aspects between the different standards [4]. In order to capitalize on the commonalities among the various signal processing operations for different standards, we need to identify firstly these commonalities and secondly find the optimal way to implement a generic hardware with reconfigurable modules. This idea led to the definition of the Common Function approach (CF) [2] which consists in function sharing between different standards. For each standard all the components dedicated to the same "Functionality" were merged into the same common function. The Common part includes the components required by at least two functions (ore function modes) and each dedicated part is related to the standard specific components of each individual function. The resource sharing brought by the CF approach allows the non-duplication of redundant components and a possible complexity reduction.

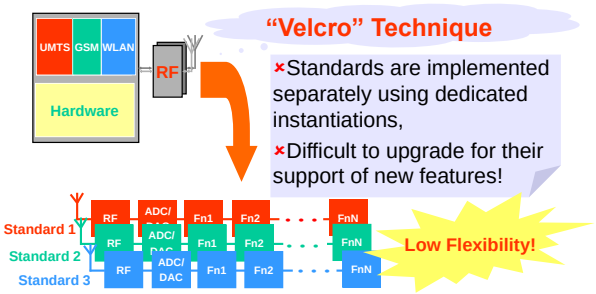

Figure 1. Velcro Technique

The Common Operator (CO) approach follows the principles that of Common Function and consists in identifying lower granularity common elements based on structural aspects. The intrinsic design of the CO is performed independently of standards. Thus, a CO is defined to perform signal processing operations regardless of the function executed. This approach aims at designing a scalable transceiver based on medium granularity operators, larger than basic logic cells and smaller than functions. In contrast with the CF, a CO is not specific to a single function set; it permits a more flexible design and scalable to a wide range of standards.

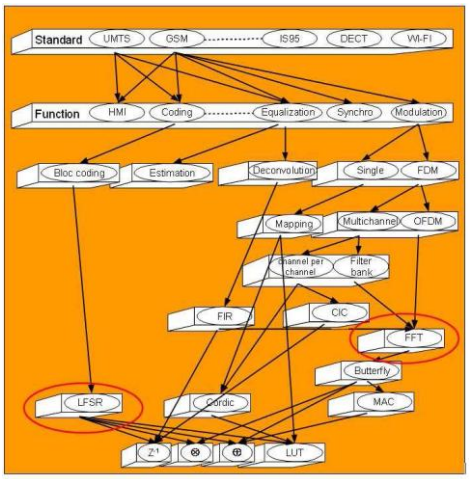

Figure 2. An example of a breakown of several standards

Fig.2 presents a graphical breakdown of a multi-standard terminal proposed in [14]. From top to bottom, the granularity of the considered components is decreased down to basic LUT or MAC. The CO consists in identifying medium granularity building blocks in such a graph to eventually address the top level functionality. The more similar the function to implement will be, the easier the identification of such blocks and the larger their granularity. For this reason, the restriction of the functional space to PHY building blocks is expected to help a lot in finding medium granularity, highly reusable operators. With a similar aim, this is one step further to identifying the Multiply ACcumulate (MAC) as a basic building block for signal processing functions.

It was shown beneficial to implement the common operators in a bank to form a regular architecture previously referred to as Common Operator Bank (COB) [8], where the COs can be mapped and used by the considered standards (Fig.3).

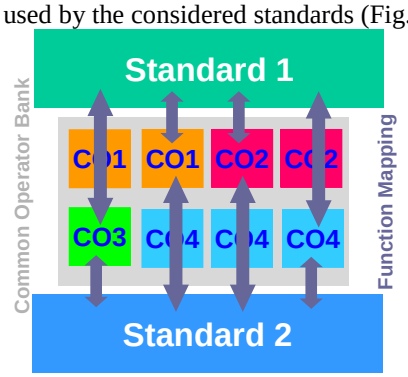

Figure 3. Common Operator Bank for multistandard design

In the present work we define common operators for FFT and FEC decoding algorithms. These algorithms are completely different in nature, if we compare their processed data and their functionality. However, when explored in the paramerization context, functional and structural similarities can be identified. In the following sections we highlight similarities between FFT and FEC decoding algorithms (Convolutional and Block channel decoding) to define a FFT/FEC CO toolbox. One can represent this way of doing by a graph sketched in Fig. 4. The interpretation of Fig. 4 is the following: performing some steps of block channel decoding (Reed Solomon) and complex FFT can be done with DMFFT operator [13]. Similarly, the proposed work intends to perform complex FFT and convolutional channel decoding thanks to a common operator termed as FFT/Viterbi.

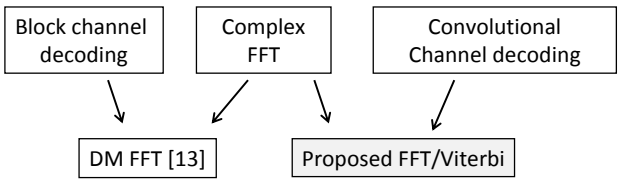

Figure 4. Common Operators graph for FFT and FEC decoding algorithms

The idea is to propose implementations of common operators that permits the use of the computational operations required for the FFT butterfly to perform Viterbi and Reed Solomon (RS) decoding.

III. FFT AND RS DECODING

In this section, the work of [16] is reminded as a first attempt to factorize FFT and FEC algorithm. More specifically, [16] focuses on the Reed-Solomon algorithm.

A. FFT over finite field

With the Fourier Transform, the concept of coding theory can be described in a setting that is much closer to the methods of signal processing. In complex field, the Fourier kernel exp(-2jπ/N) is an N^{th} root of unity in the field of complex numbers. In the finite field $GF(q)$ an element α of order N is an N^{th} root of unity. Drawing on the analogy between exp(-2jπ/N) and α, Fourier transform over finite field can be defined as follows [24]: let $f=(f_0, f_1, ..., f_{N-1})$ be a vector over $GF(q)$, and let α be an element of $GF(q)$ of order N. The Fourier transform of vector f is the vector $F=(F_0, F_1, ..., F_{N-1})$ whose components are given by

$$F_j = \sum_{i=0}^{N-1} f_i \alpha^{ij}, \quad j = 0,..., N-1. \tag{1}$$

Vector f is related to its spectrum F by

$$f_i = \frac{1}{N} \sum_{j=0}^{N-1} F_j \alpha^{-ij}, \quad i = 0,..., N-1. \tag{2}$$

It is natural to call the discrete index i '*time'*, taking values on the *time axis* 0, 1, ..., N-1, and to call f the '*time-domain function'* or the *'signal'*. Also, one might call the discrete index '*frequency'*, taking values on the frequency axis 0, 1, ..., N-1, and to call F the '*frequency-domain'* or the '*spectrum'*.

Fourier transform in Galois field [27] closely mimics the Fourier transform in the complex field with one important difference: in the complex field an element W of order N (e.g. exp(-2jπ/N)), exists for every value of N but in $GF(q)$, such an element W exists only if N divides q-1. Moreover, if for some values of m, N divides q^m-1 then there will be a Fourier transform of length N in the extension field $GF(q^m)$. For this reason, a vector f of length N over $GF(q)$ will also be regarded as a vector over $GF(q^m)$ and has a Fourier transform of length N over $GF(q^m)$. This is completely analogous to the Fourier transform of a real-valued vector: even though the time-domain vector f has components only in the real field, the transform F has components in the complex field. Similarly, for the finite Fourier transform, even though the time-domain vector f is over the field $GF(q)$, the spectrum F may be over the extension field $GF(q^m)$. Any factor of q^m-1 can be used as the length of a Fourier transform over $GF(q)$, but the most important values for N are the primitive length $N=q^m$-1. In that case $W = \alpha$ is a primitive element of $GF(q^m)$.

B. FFT and RS Common Operator

The most popular class of RS cyclic codes are defined over $GF(q=2^m)$. However the transform length of the finite field transform over $GF(2^m)$ equal to 2^m-1 does not match the one of the complex FFT defined over the complex field, which is 2^m. This characteristic is a strong constraint that challenges the adaptation or the combination of the $GF(2^m)$ FFT structure with the complex FFT one, since most efficient algorithms regarding FFT computations are applied to transforms of length 2^m. Under this strong constraint, one thought to seek out a transform matching these complex FFT criteria. State of the art on finite field transforms and RS codes leads to spot specific class of transforms and get out the corresponding class of RS codes [16]. These specific finite field transforms as well as the corresponding RS codes are defined over $GF(F_t)$ where F_t is a Fermat prime number defined as $F_t = 2^{2^t} + 1$. Fourier transform defined over this specific Galois field $GF(F_t)$ known as Fermat Number Transform (FNT) can play a leading role in the frequency processing of RS codes: the encoding and the most important tasks of RS decoding (i.e. *syndrome computation* and *Chien search*) and can be performed with FNT.

Hardware realization of the common operator can be now presented to perform with the same architecture Fourier transforms over $GF(F_t)$ and over **C.**

The classical complex FFT architecture is re-design in a way to enable to perform the FNT. A radix-2 FFT implementation is considered because it has advantages in terms of regularity of hardware, ease of computation and number of processing elements. Obviously, for a given transform length N power of 2 (or power of 4), the algorithm chosen to be applied to perform FFT should be valid to perform the FNT. Indeed, since the symmetry and periodicity properties $\alpha^{K+N} = \alpha^k$ and $\alpha^{K+N/2} = -\alpha^k$ are verified, every radix-2 algorithm applied to FFT can be applied to the FNT. The heart of this algorithm known as the "butterfly" was re-designed. Here re-designing means taking into account the re-configuration of the operators constituting the butterfly as well as the connection between those operators. The switching from FFT mode to FNT mode should be accompanied by the replacement of the twiddle factor W by the primitive element α of the given Galois field.

In the FFT mode these operators process complex data by performing complex multiplications and additions. In the FNT mode data are defined over finite field and the operations performing FNT are done modulo F_t. So, these arithmetic operators should be re-redefined to support complex and modular operations. Fig. 5 illustrates the associated butterfly operator architecture.

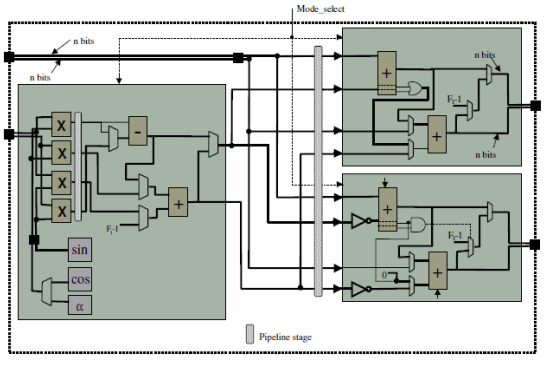

Figure 5. FFT/RS butterfly Common Operator

IV. FFT AND VITERBI DECODING

With a similar aim, the new CO presented herein intends to address the infinite field FFT and the Viterbi decoding algorithms. First, the algorithms are analyzed to highlight the similarities between the Treillis and the Butterfly structures which are further exploited to build the new CO.

A. The FFT butterfly

The FFT over infinite field is an efficient algorithm to compute the discrete Fourier transform (DFT). The DFT is defined by the equation (3) where x_0, ..., x_{n-1} are complex numbers.

$$X_j = \sum_{k=0}^{n-1} x_k . e^{-\frac{2.\pi.i}{n}.k}, \; j = 0,\ldots,n-1 \tag{3}$$

The term "butterfly" is commonly quoted in the Cooley–Tukey FFT algorithm context [9], which recursively decomposes the DFT of size n = r×m into r (radix) smaller transforms of size m. These smaller DFTs are then combined with butterflies of size r, which themselves are DFTs of size r pre-multiplied by twiddle factors. The term “butterfly” comes from the shape of data-flow diagram in the radix-2 algorithm. In this paper we consider the most popular Cooley-Tukey FFT which is the radix-2 case.

The radix-2 FFT algorithm is usually applied when the FFT size is a power of 2. Equation (4) shows that at the k^{th} step the results of two smaller Fourier transform are needed. Then, using the divide and conquer strategy, a k-point transform can be reduced to two k/2-point transforms: one for even samples, one for odd samples (Fig.6).

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x_n . e^{-\frac{2.\pi.i}{N}.n.k} \\ &= \sum_{m=0}^{N/2-1} x_{2m} . e^{-\frac{2.\pi.i}{N}.2.m.k} + \sum_{m=0}^{N/2-1} x_{2m+1} . e^{-\frac{2.\pi.i}{M}.(2.m.+1).k} \end{aligned} \tag{4}$$

Starting with N that is a power of 2, it is possible to apply this subdivision recursively until getting down to 2-sample transforms that can be represented graphically using the, so called, butterfly in Fig.7.

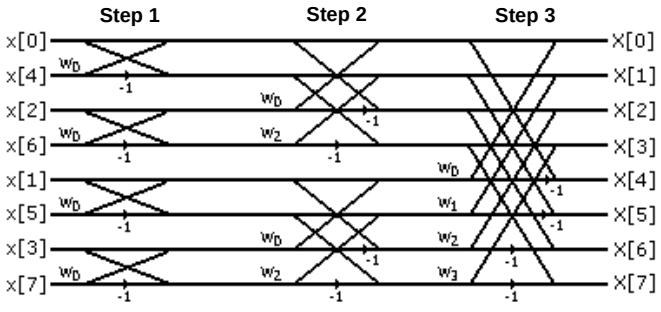

Figure 6. Treillis based FFT algorithm

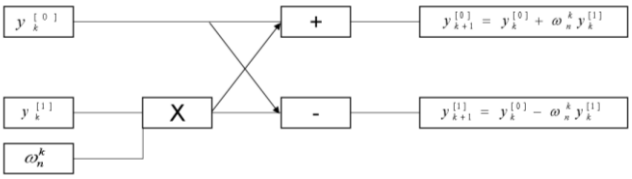

Figure 7. Radix-2 FFT butterfly structure

B. The Viterbi Butterfly

The same structure can also be identified in the Viterbi algorithm, used for finding the most likely sequence of states in a trellis, which is the most usual representation of convolutional code state diagram. Although it is not the most compact form, the trellis structure is commonly used because it easily illustrates the sequencing of decoding algorithms (Fig.8).

The classical architecture of the Viterbi algorithm can be divided into three units, as shown in Fig.9. The Branch Metric Calculation (BMC) unit computes the distances (branch metrics) associated to each transition of the trellis in order to evaluate the correctness of the received data for a given transition. Secondly, the Add Compare Select unit (ACS) computes the accumulated metrics (called path metrics) and selects the incoming survivor path for each state of the trellis. Finally, the Survivor Memory Management unit (SMM) stores the decision taken by the ACS unit in order to provide the most likely decoded path at the output of the decoder.

By focusing on the butterfly structure of the Viterbi algorithm, it is possible to analyze how it operates and then highlight the similarities with the FFT butterfly. Indeed, the Viterbi butterfly involves not only the computation of the path metrics, but also a comparison module. The comparison module can easily be realized using a subtractor associated to a multiplexor. Thus, for the implementation of the path metrics computation, two adders and one subtractor are required. Equation (5) describes the computation performed by the butterfly and Fig.10 illustrates its structure. The defined Viterbi butterfly corresponds to the ACS module in Fig. 9.

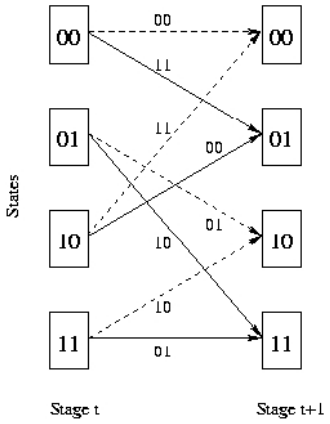

Figure 8. Four-state Trellis diagram for Viterbi decoding

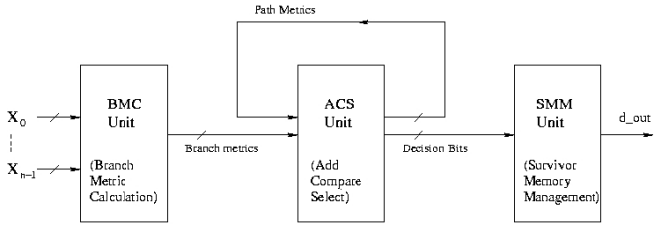

Figure 9. Viterbi decoder structure

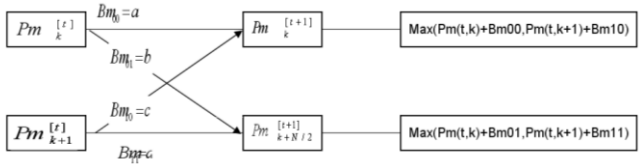

Figure 10. Viterbi decoder butterfly structure

$$Pm_k^{[t+1]} = Max(Pm(t,k) + Bm00, Pm(t,k+1) + Bm10)$$
$$Pm_k^{[t+1]} = \begin{cases} Pm(t,k) + Bm00, \ldots if \ldots Pm(t,k) + Bm00 - Pm(t,k+1) + Bm10 > 0 \\ Pm(t,k+1) + Bm10, \ldots if \ldots Pm(t,k) + Bm00, Pm(t,k+1) + Bm10 < 0 \end{cases} \quad (5)$$

As shown in this section, the FFT and Viterbi algorithms have strong similarities if we compare their butterfly structures. These similarities can be explored to build a common structure for the two algorithms.

C. Proposed FFT/Viterbi Common Operator

In this section, we a build a common structure for the FFT and Viterbi decoding algorithms starting from the architectures of the previously presented FFT and Viterbi decoder.

1) FFT butterfly structure

From the FFT butterfly (Fig.7), we define the computation method using bit-parallel multipliers for complex-valued operations [11]. In Equation (6) we define real and complex parameters for the FFT butterfly.

$$\begin{aligned} y^{[1]}{}_k &= a + jb \\ w^k{}_n &= c + jd \\ y^{[0]}{}_k &= e + jf \end{aligned} \quad (6)$$

After developing the complex operations required for the butterfly computation, Equation (7) is obtained.

$$\begin{cases} y^{[0]}{}_k + y^{[1]}{}_k * w^k{}_n = e + (ac - bd) + j.[f + (ad + bc)], \\ y^{[0]}{}_k - y^{[1]}{}_k * w^k{}_n = e - (ac - bd) + j.[f - (ad + bc)] \end{cases} \quad (7)$$

This "direct" form of the FFT butterfly computation requires four multipliers and six adders. The number of multipliers can be reduced by rewriting Equation (7) into a different form as shown in Equation (8).

$$\begin{cases} y^{[0]}{}_k + y^{[1]}{}_k * w^k{}_n = [e + (c.(a+b) - b.(c+d))] + j.[f + (c.(a+b) + a.(d-c))], \\ y^{[0]}{}_k - y^{[1]}{}_k * w^k{}_n = [e - (c.(a+b) - b.(c+d))] + j.[f - ((c.(a+b) + a.(d-c))] \end{cases} \quad (8)$$

Based on the previous equations, we propose a structure for the FFT butterfly using three multipliers (Fig.11).

2) Viterbi butterfly structure

Starting from (5), we can deduce the operations required by the Viterbi butterfly implementation. The computation of every path metric requires tow adders and one subtractor as illustrated in Fig.12.

Branch metrics are evaluated in the BMC block. At the output of this block, the difference between the received value and the different transitions related to it are evaluated. The computed metrics are then distributed to all the butterflies of ACS module. Thus the recalculation of the butterfly parameters requiring the same metrics is avoided. The branch metric computation can be designed with simple addition and subtraction operations between the decoder inputs. Thus, for R soft received inputs, all possible metric consists in 2^R possible operation (addition or subtraction) which can be reduced by half, since half of the metric can be deduced from the other half by a simple change of sign. Indeed, the 2^{R-1} first metrics are computed from R soft inputs, and the last 2^{R-1} metrics are evaluated from the first ones by a simple change of sign.

Then, the execution of every butterfly requires the branch metrics Bm00, Bm01, Bm10 and Bm11, and the whole Viterbi Butterfly computation (ACS+BMC) can be realized as shown in Fig.13.

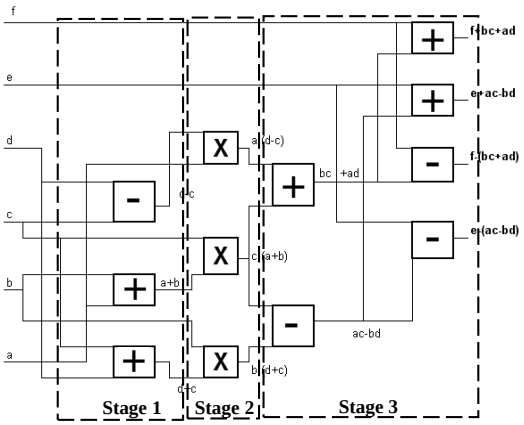

Figure 11. Three-multipliers based FFT butterfly implementation

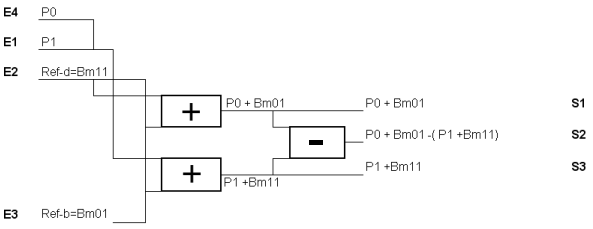

Figure 12. Viterbi butterfly implementation for path metrics evaluation

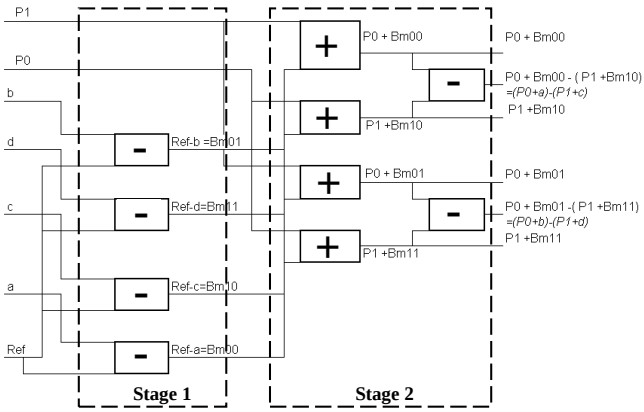

Figure 13. Full Viterbi butterfly (BMC + ACS) implementation

3) FFT and Viterbi common structure

Starting from the previously presented structures of the FFT and Viterbi butterflies, we propose in this paragraph a Common Operator that addresses the requirements of the two algorithms. This architecture can perform the calculation of the FFT butterfly and the (BMC + ACS) operations of the Viterbi decoding algorithm.

As presented in Fig.11, the FFT butterfly requires nine add/subtract (A/S) operations and three multiplications. We can split them it into three stages: the first stage is composed by three A/S, three multipliers for the second stage and the last stage is composed by 2×3 A/S operators as shown in Fig. 14. On the other hand, he Viterbi module (BMC+ACS) can be divided into two stages as illustrated in Fig.13. The first stage is composed of four A/S (BMC stage) and 2×3 A/S for the second stage (ACS stage).

This decomposition allows the pooling of the FFT and Viterbi operators by suggesting a common structure for each stage. The first stage for each operator requires independent add/subtraction operations, four for the Viterbi BMC (or two for a more optimized form) and three for the FFT. Then the first stage of the common operator consists of three A/S blocks. The second stage is dedicated to the FFT butterfly, it consists in three multipliers.

Unlike the two previous stages, the adders and subtructors in the third stage are interconnected and are dependent on each others. In this stage, six A/S are required but interconnected by three for real and imaginary parts of the FFT butterfly. As illustrated in the Fig.11 and Fig.13, the A/S blocks are not interconnected in the same way for the two algorithms. Thus, in order to build a common structure for this stage we develop the equations below to show a common mathematical expression for the third stage of the Viterbi and FFT butterflies.

Fig. 11 and Fig.13, actually perform Equations (7) and (8) linking the inputs and outputs of the third stage.

For the imaginary part of the FFT butterfly and the first Viterbi butterfly:

$$\begin{cases} S1 = (1-\beta).E3 + \beta.S3 + E4 = \beta.(E1+E2) + (1-\beta).E3 + E4 \\ S2 = [\beta.E4 + (1-\beta).S1] - S3 = \beta.E4 + (1-\beta).(E3+E4) - (E1+E2) \\ S3 = E1 + E2 \end{cases} \quad (7)$$

For the real part of the FFT butterfly and the second Viterbi butterfly:

$$\begin{cases} S1 = (1-\beta).E3 + \beta.S3 + E4 = \beta.(E1+(-1)^{\beta}.E2) + (1-\beta).E3 + E4 \\ S2 = [\beta.E4 + (1-\beta).S1] - S3 = [\beta.E4 + (1-\beta).(E3+E4)] - [E1+(-1)^{\beta}.E2] \\ S3 = E1 + (-1)^{\beta}.E2 \end{cases} \quad (8)$$

β is a boolean parameter (implemented through a single bit selector) that permits the configuration of the common structure to switch between the FFT and Viterbi computation. In Fig.15 we present the graphic representation of the previously developed equations.

Starting from the previous discussion and by gathering the developed stages, the entire FFT and Viterbi Common Operator is presented in Fig.16. This common operator architecture allows switching between two different functional implementations of the Viterbi and FFT algorithms. The reconfiguration can be easily performed using a single parameter. The reconfigurable operators are composed by real adders and multiplexers.

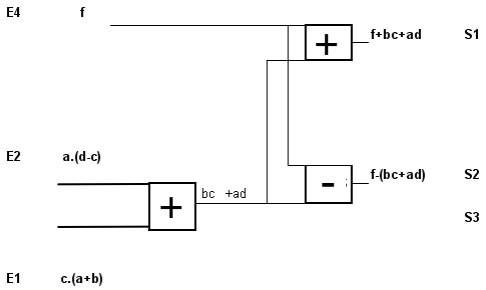

Figure 14. Third stage of the FFT butterfly decomposition

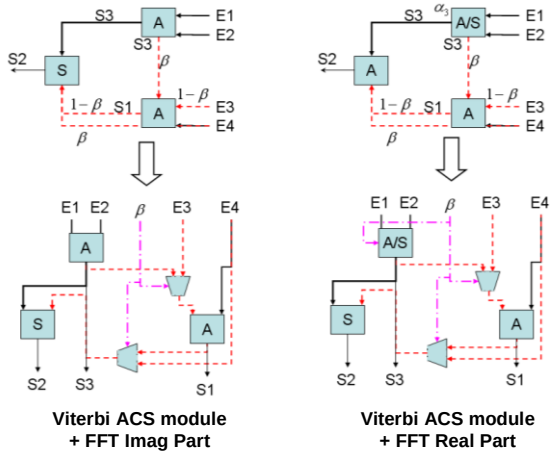

Figure 15. Third stage for the FFT/Viterbi Common structure

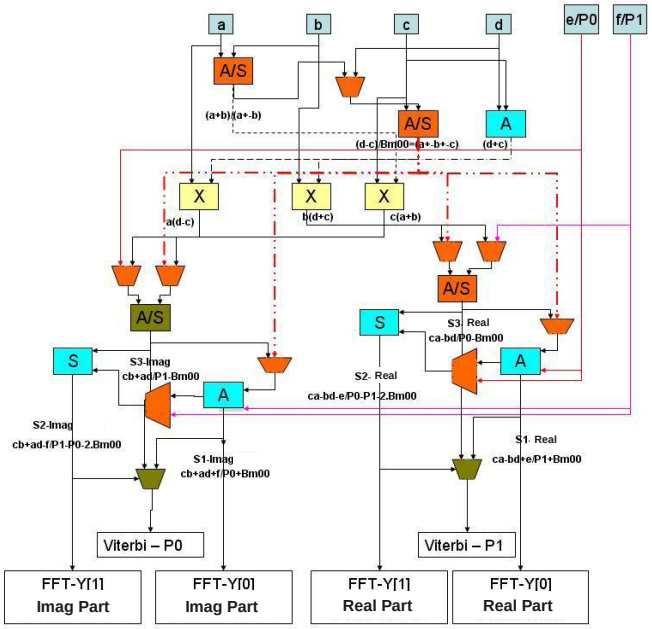

Figure 16. FFT/Viterbi Common Operator

V. Complexity Evaluation

In order to evaluate the performance and complexity of the finite/infinite field Dual Mode FFT (DMFFT) architecture, it was implemented on FPGA and compared to a Velcro FFT/FNT operator implemented on the same target device. Complexity evaluations in [13] showed that depending on the word-length n_c, DMFFT exhibits a memory saving between 20 and 30 %, a gain in ALUTs and performance-to-cost ratio gain from 9.2 % up to 26 % and from 9.7 % up to 37.4 % respectively [13].

With the aim of analyzing the impact of the FFT/Viterbi CO on the global implementation complexity, we explored the possible design scenarios considering the number of the physically instantiated butterflies in a multistandard design [28]. Then, the global complexity reduction is evaluated by comparing the number of implemented logic gates for the CO and Velcro based designs.

To explain this comparison, we illustrate in Fig. 17 a simplified example of a two-standard terminal. For a classical implementation (Velcro), we need to implement the maximum number of required FFT butterflies and the maximal number of Viterbi butterflies. On the other hand, the use of the CO enables to reduce the number implemented butterflies because of the 'reuse' across the algorithms, rather than just across standards.

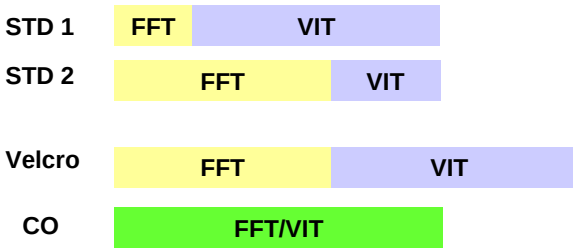

Figure 17. An example showing resource allocation reduction for a CO based two-standard implementation

In this case, the use of the FFT/Viterbi common operator can reduce the complexity of the FFT and Viterbi by up to 5%, when the number of the implemented Viterbi butterflies is equal to the number of implemented FFT butterflies [28]. On the other hand, the use of the DMFFT operator can reduce the design complexity by up to 26% compared to classical implementations [13]. This complexity gain can be interpreted as little, but it should be kept in mind that the main motivation of this work is to build operators of higher flexibility and that can be used in a regular architecture. For that reason, showing that this additional flexibility is not traded against additional complexity is a very promising result.

In Table I we compare the proposed FFT/Viterbi common operator with other reconfigurable PEs [25,26]. As shown in this Table, the proposed cell provides an important gain in complexity compared to [26] with a good performance in terms of number of operations per cycle compared to [25]. Also throughput reduction can be compensated by the reuse of physical entities through operator time multiplexing. Indeed, [10] has shown that limiting the number of physical butterflies' instances can be achieved without significant overhead from the time multiplexing management.

TABLE I
Performance Comparisons for the Considered Operations

		Proposed FFT/Viterbi CO	**RMAC-PE [25]**	**RCC [26]**
Complexity (Gate Count)		Reference	-26%	+103%
Operations/Cycle	**FFT Radix-2 Butterfly**	1	0,33	1,5
	Path Metrics Calc. (ACS)	2	0,5	2
	Branch Metrics Calc. (BMC)	1	0,33	4

VI. Conclusion

In this paper, the similarities between the FFT and two FEC decoding algorithms were studied. Based on the parametrization technique, a pooling method focusing on FFT/Viterbi and FFT/RS butterflies' structure were presented.

Common Operator architectures that can be used in the FFT, Viterbi and RS decoding were proposed. It was shown that with this CO it is possible to build a more general library framework for FFT/FEC functions that permits. The proposed regular structures can bring considerable advantages when implemented in a Common Operator Bank (COB) [8]. In addition, this gain in flexibility is done at no overhead cost since the complexity is even decreased in some configurations.

REFERENCES

[1] J. Mitola, "The software radio architecture," IEEE Communications Magazine, 33, pp. 26–38, 1995.

[2] F. Jondral, "Software Defined Radio: Enabling technologies", Wiley 2002.

[3] W. Tuttlebee, "Software Defined Radio – Baseband Technology for 3G Handsets and Basestations," Communications Engineer vol. 2, no. 2, pp. 46-47.

[4] L. Alaus, J. Palicot, C. Roland, Y. Louet, D. Noguet, "Promising Technique of Parameterization For Reconfigurable Radio, the Common Operators Technique: Fundamentals and Examples" Journal of Signal Processing Systems, Springer, 2009.

[5] F. Jondral, "Parameter Controlled Software Defined Radio," Software Defined Radio Technical Conference, San Diego, Nov 2002.

[6] D. Noguet, G. Masera, V. Ramakrishnan, M.Belleville, D. Morche, G. Asheid, "Considering microelectronic trends in advanced wireless system design" Advances in Electronics and Telecommunications Journal, April 2010.

[7] C. Moy, J. Palicot, V. Rodriguez, D. Giri, "Optimal Determination of Common Operators for Multi-Standard Software-Defined Radio", 4th Karlsruhe Workshop on Software Radios, March 2006.

[8] L. Alaus, D. Noguet, J. Palicot, "A Common Operator Bank to resolve scheduling issue on a SDR Terminal". The 6th Advanced International Conference on Telecommunications, 2010. [Submitted]

[9] Cooley, J. W. and Tukey, J. W, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Computat., 19, 297–301, 1965

[10] L. Biard, D. Noguet, "An adaptable architecture for the Viterbi algorithm", The 7th International Symposium on Wireless Personal Multimedia Communications, 2004.

[11] J.Takala, K. Punkka, "Scalable FFT Processors and Pipelined Butterfly Units", The Journal of VLSI Signal Processing, Springer Netherlands 2006

[12] S.T. Gul, C. Moy, J. Palicot, "Two scenarios of flexible Multi-standard architecture designs using a multi-granularity exploration", The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2007

[13] Ali Al Ghouwayel and Yves Louet "FPGA implementation of a re-configurable FFT for multi-standard systemes in software radio context", IEEE Trans. On Consumer Electronics Journal, vol. 55, n°2, pp. 950-958, May 2009

[14] Rodriguez, V., Moy, C., Palicot, J. (2007). Install or invoke?: The optimal tradeoff between performance and cost in the design of multi-standard reconfigurable radios. In Wiley InterScience, Wireless Communications and Mobile Computing Journal 7(9), (pp. 1143–1156), DOI 10.1002/wcm.487.

[15] J. Palicot, C. Roland, "FFT: a Basic Function for a Reconfigurable Receiver" ICT' 2003, Papeete, Tahiti.

[16] A. Al Ghouwayel, Y. Louët and J. Palicot, "A Reconfigurable Architecture for the FFT Operator in a Software Radio Context", IEEE ISCAS'2006, Greece, May 2006.

[17] W. C. Gore, Transmitting Binary Symbols with Reed-Solomon Codes, Proceedings of Princeton Conference on Information Sciences and Systems, Princeton, NJ, 1973, pp. 495-497.

[18] A. Michelson, A Fast Transform in Some Galois Field and an application to Decoding Reed-Solomon Codes, IEEE International Symposium on Information Theory, Ronneby, Sweden, 1976, p. 49.

[19] A. Lempel and S. Winograd, A New Approach of Error Correcting Codes, IEEE Trans. Inf. Theory IT-23, 503-508, 1977.

[20] R. T. Chien and D. M. Choy, Algebraic Generalization of BCH-Goppa-Helgert Codes, IEEE Trans. Inf. Theory IT-21, 70-79, 1975.

[21] S. M. Reddy and J.P. Robinson, Random Error and Burst Correction by Iterated Codes, IEEE Trans. Inf. Theory, vol IT-18, p. 172-185, Jan. 1972.

[22] S. B. Wicker and V. K. Bhargava, Reed-Solomon codes and their applications, New York: IEEE Press, 1994.

[23] S. B. Wicker and V. K. Bhargava, Reed-Solomon codes and their applications, New York: IEEE Press, 1994.

[24] A. Rhiemeier, Benefits and limits of parameterized channel coding for software radio, 2nd Workshop on Software Radio, Karlsrhue, Germany, March 2002. A. Rhiemeier, Benefits and limits of parameterized channel coding for software radio, 2nd Workshop on Software Radio, Karlsrhue, Germany, March 2002.

[25] H. Lange, O. Franzen, H. Schröder, M. Bücker, B, Oelkrug, "Reconfigurable Multiply-Accumulate-based Processing Element", In: Proc. of the IEEE Workshop on heterogeneous Systems on a Chip, Hamburg, Germany, 2002.

[26] C.Y. Jung, M.H. Sunwoo, S.K. Oh, "Design of reconfigurable coprocessor for communication systems", SIPS 2004. IEEE Workshop on Signal Processing Systems, 2004.

[27] J. M. Pollard, The fast Fourier transform in a finite field, IEEE Trans. Comput., vol. 25, pp. 365-374, Apr. 1971.

[28] M. Naoues, L. Alaus, D. Noguet, "A Common Operator for FFT and Viterbi Algorithms," Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on , vol., no., pp.309-313, 1-3 Sept. 2010 doi: 10.1109/DSD.2010.80

***Biography of all Authors**
Click here to download Biography of all Authors: bio-1.doc

Malek Naoues received his Diploma of Engineer in Telecommunications from Higher School of Communication of Tunis in 2008 and his MSc in Telecommunication from the same school in 2009. Since 2010, he works for CEA-LETI as a Ph.D student in collaboration with SUPELEC-France and SUP'Com-Tunisia. His research activities regard flexible architectures for Multi-standards terminals.

Dr. Dominique NOGUET graduated from the National Institute of Applied Sciences (INSA) in electrical engineering in 1992. He obtained an MSc in microelectronics of the University of Strasbourg in 1994 and a PhD of Polytechnic National Institute Grenoble (INPG) in 1998 (Awarded "best INPG PhD"). Since then, he has been with CEA-LETI where he has worked as ASIC designer, system architect, and project manager in wireless digital processing. He has coordinated several national and European research projects, among which ORACLE, the first EU project on Opportunistic Radio. He is currently the technical manager of the EU QoSMOS project on cognitive radio, and the head of the "digital architectures and prototypes" ANP group at CEA-LETI, where he also leads Cognitive Radio Activities. He has been appointed "senior expert" of CEA since 2007. His main fields of interest are on flexible digital radios and cognitive radios.

Dr. Laurent Alaus was born in 1982. He received his Master Degree of Science in 2006 from INSA Lyon, France, where he dedicated his first research to cyclostationarity detection schemes for Cognitive Radio. From 2007 to 2010, he worked for CEA-LETI, France in collaboration with SUPELEC-SCEE Laboratory on flexible architectures for multi-standards terminals and received his Ph.D. degree in Telecommunications in May 2010.

Dr. Yves LOUËT (M'04) was born in 1973. He received his Ph.D. degree in Digital Communications in 2000 and his Research Habilitation in 2010 from Rennes University, France. He is Professor in SUPELEC- France and his research activities regard signal processing and digital communications for Software Radio systems.