Low Complexity FFT/IFFT Processor Applied for OFDM Transmission System in Wireless Broadband Communication

M. Arioua, *Member, IACSIT*, and M. M. Hassani

Abstract—Processors of Fast Fourier Transform (FFT) and its inverse transform IFFT are a key component in OFDM based wireless broadband communication systems. It is essential to develop a high-speed and low-power performance FFT/IFFT processor to meet the real time and low cost prerequisites in such communication systems. This paper gives the details of the development of efficient 256-points FFT/IFFT architecture to be used in OFDM system and which satisfies the specification of IEEE 802.16a standard. The 256-points FFT architecture consists of an optimized pipeline implementation of 16-point FFT core based on Radix-2 butterfly Processor Element. This proposed design reduces the multiplicative complexity compared to other efficient architectures. The FFT processor has been implemented in VHDL code, and simulation results show that this FFT module significantly achieves a better performance with low arithmetic complexity. Hence high speed and low power consumption for OFDM-based wireless broadband communication systems.

Index Terms—FFT/IFFT, FPGA, IEEE 802.16a, MDC, OFDM, Radix-2, SDF, wireless broadband communication.

I. INTRODUCTION

Currently, one of the most commonly used digital signal processing algorithm is Fast Fourier Transform (FFT) and its inverse transform (IFFT). Recently, this algorithm has been widely used in digital signal processing field applied for wireless communication systems. FFT/IFFT is a key component of the physical layer of Orthogonal Frequency Division Multiplexing (OFDM) based wireless broadband communication system; it's one of the most complex and intensive computation module of various wireless standards PHY layer (OFDM-802.11x, MIMO-OFDM 802.11n, OFDM-802.16x) [1].

The fast growing demand of OFDM-based applications, including Wireless Metropolitan Area Network (WMAN) applications, makes the processing speed an indicative factor in the Fast Fourier Transform algorithm design. Indeed, these applications need the FFT and IFFT processors to perform real-time operations for modulation and demodulation of OFDM signals. Hence, the study of FFT/IFFT algorithms and high performance VLSI FFT/IFFT architecture becomes increasingly important [2].

The main constraints nowadays for FFT/IFFT processors used in IEEE 802.11x and IEEE 802.16x standard, are execution time and power consumption [2], [3]. According to the 802.16a specifications, the 256-point IFFT/FFT has to perform FFT computation in 89.6 μ s. To fulfill this time constraint, the FFT design has to use a highly in-depth architecture or employ a very high operation frequency. Both solutions will satisfy the timing constraint, but at the cost of the area and power consumption. In this paper, we concentrate on an optimal implementation on Altera FPGA platform of the 256-point FFT processor based on 16-points FFT core. We propose a power efficient 256-points FFT architecture that has low arithmetic complexity and high architecture regularity and at the same time satisfies the timing of the IEEE 802.16a specification.

The paper is structured as follows: Section II discusses the Cooley-Tukey algorithms and complex multiplication inside a butterfly-Processing Element. Section III discusses the mathematical formulation and architectural description of the FFT proposed processor. Section IV shows the resulting implementation and the performance evaluation. Finally a conclusion is given in the last section.

II. FFT/IFFT ALGORITHMS

A. Cooley-Tukey Algorithms

The N-point Discrete Fourier Transform (DFT) of complex data sequence $x(n)$ is defined as:

$$X[k] = \sum_{n=0}^N x[n] \cdot W_N^{nk} \quad (1)$$

where: $W_N^{nk} = e^{-j \frac{2\pi nk}{N}}$ and $0 \leq k \leq N-1$

$X(k)$ is the k-th harmonic and $x(n)$ is the n-th input sample. Direct DFT calculation requires a computational complexity of $O(N^2)$. By using The Cooley-Tukey FFT algorithm, the complexity can be reduced to $O(N \log_r N)$ [4].

The most universal of overall FFT algorithms is Cooley-Tukey, because of any factorization of N is possible [5]. The most popular Cooley-Tukey algorithms are those where the transform length is a power of a basis r , i.e., $N=r^S$. These algorithms are referred to as Radix-r algorithms. The most commonly used are those of basis $r=2$ (radix-2) and $r=4$ (radix-4), $r=8$ (radix-8) and $r=16$ (radix-16). Those algorithms and others such as radix-2², radix-2³, Split-Radix have been developed based on the basic Cooley-Tukey algorithm to

Manuscript received August 25, 2013; revised November 18, 2013

M. Arioua is with the National School of Applied Sciences, University of Abdelmalek Essaadi Tetouan-Tanger, B.P.2222 M'hannech, Morocco (email: m.arioua@ieee.org).

M. M. Hassani is with the Faculty of Science of Marrakech at Cadi Ayyad University, B.P.511- 40000, Morocco (email: hassani@ucam.ac.ma).

further reduce the computational complexity [6].

For $r=2$ and S stages, for instance, the following index mapping of Cooley–Tukey algorithm gives:

$$n = 2^{S-1}n_1 + 2^{S-2}n_2 + \dots + 2n_{S-1} + n_S \quad (2)$$

$$k = 2^{S-1}k_S + 2^{S-2}k_{S-1} + \dots + 2k_2 + k_1 \quad (3)$$

And $n_1, n_2, \dots, n_{S-1}, n_S = 0, 1$, $k_S, k_{S-1}, \dots, k_2, k_1 = 0, 1$

This algorithm is based on a divide-and-conquer approach in the frequency domain and therefore, is referred to as decimation-in-frequency (DIF) FFT. The FFT formula is split into two summations:

$$X(k) = \sum_{n=0}^{N/2-1} x(n)W_N^{nk} + \sum_{n=N/2}^{N-1} x(n)W_N^{nk} \quad (4)$$

$$X(k) = \sum_{n=0}^{N/2-1} \left(x(n) + (-1)^k x\left(n + \frac{N}{2}\right) \right) W_N^{nk} \quad (5)$$

After decimation into even-and odd-indexed frequency samples, $X(k)$ becomes:

$$X(2k) = \sum_{n=0}^{N/2-1} \left(x(n) + x\left(n + \frac{N}{2}\right) \right) W_N^{2nk} = \sum_{n=0}^{N/2-1} \left(x(n) + x\left(n + \frac{N}{2}\right) \right) W_{N/2}^{nk} \quad (6)$$

$$X(2k+1) = \sum_{n=0}^{N/2-1} \left(x(n) - x\left(n + \frac{N}{2}\right) \right) W_N^{(2k+1)n} = \sum_{n=0}^{N/2-1} \left(x(n) - x\left(n + \frac{N}{2}\right) \right) W_{N/2}^{nk} W_N^n \quad (7)$$

The computational method can be repeated through decimation of the $N/2$ -point FFTs $X(2k)$ and FFTs $X(2k+1)$. The entire algorithm involves $\log_2 N$ stages, where each stage involves $N/2$ operation units (Butterflies). The computation of the N point FFT via the decimation-in-frequency (DIF) as in the decimation-in-time (DIT) algorithm requires $(N/2) \cdot \log_2 N$ complex multiplications and $N \cdot \log_2 N$ complex additions/subtractions [7]. Based on the same approach, the other fast algorithms: radix-4, radix-8, radix-16, radix- 2^2 and split-radix recursively divide the FFT computation into odd and even-half parts and then obtain as many common twiddle factors as possible. The number of needed real additions and multiplications is generally used to compare efficiency of different variants of FFT algorithms.

The split-radix algorithm offers the best computational performance, as indicated by multiplicative comparison in [8], due to its most trivial multiplication with twiddle factors W_N , i.e., ± 1 and $\pm j$. Nevertheless, the split-radix is an irregular algorithm by its nature, because of the combination of two algorithms radix-2 and radix-4 stages used respectively for the even-half operations and for the odd half operations. Therefore, this architecture leads to an L-shaped butterfly units and affect the delay of the pipeline path and make it unbalanced [6], [8]. A fix radix algorithm with low degree, as radix-2, is entirely convenient for its requirement of low complexity of integration in an integrated circuit(IC), due to the algorithm regularity as well as the conception complexity and architecture control.

Fig. 1 represents the flow graph of complete decimation-in-frequency decomposition of 16-points FFT computation based on radix-2. The intrinsic operation of the signal flow graph is the butterfly operation; it's a 2-point DFT computation.

B. 16-Points FFT/IFFT Module

Fig. 1. 16-Points DIF FFT algorithm based on Radix-2.

The FFT computation of 16-points radix-2 based architecture is achieved in four stages. The $x(0)$ until $x(15)$ variables are indicated as the input values for FFT computation and $X(0)$ until $X(15)$ are indicated as the output values. In the butterfly process, the upward arrow executes addition operation, beside that; downward arrow executes subtraction operation. The subtracted value is multiplied with twiddle factor value W_N before being processed into next stage; this computation performed concurrently. The complex multiplication with the twiddle factor requires four real multiplications and two add/subtract operations [4], [7].

The complex multiplication is one of the most essential arithmetic operations used in FFT computation. It is often the most expensive arithmetic operation and one of the dominate factors in determining the performance in terms of power consumption, speed and throughput of an FFT processor [9].

As observed in [10], the complex multiplier may consume more than 70% of the power in an FFT/IFFT processor. Therefore, an effective design of FFT processor is vital in high speed and low-power applications.

The aim here is to reduce the multiplication complexity of the twiddle factor inside the butterfly processor by calculating only three real multiplications and three additions/subtractions operations [4], [11]. This method of complex multiplication reduction is demonstrated in equation 8 and equation 9. We applied it for efficient conception of 16-points FFT module.

The complexes twiddle factor multiplication:

$$R + jI = (X + jY) \cdot W_N = (X + jY) \cdot (C + jS) \quad (8)$$

However, it can be simplified:

$$R = (C - S) \cdot Y + Z \quad (9)$$

$$I = (C + S) \cdot X - Z \quad (10)$$

and

$$Z = C \cdot (X - Y) \quad (11)$$

The twiddle factors coefficients W_N are known in advance depend on the algorithm adopted in the FFT implementation. i.e., C and S in Equations 9 and 10 are pre-computed and

stored in a memory table. Therefore it is necessary to store the following three coefficients C , $C+S$, and $C-S$. The storage operation of those constants is used to simplify the complex multiplication. Those constants can be saved as canonical signed digits (CSD) to implement complex multiplication with carry and save tree [12]. Consequently, the area and power consumption can both be reduced.

The complex multiplication with $W_{16}^2 = e^{-j\frac{\pi}{4}}$ requires only two real multiplications rather than three multiplications. Moreover, the complex multiplication can be reduced further with an efficient number representation of fixed-point arithmetic [13]. The implementation of 16-points FFT radix-2 algorithm is accomplished by coding the FFT module in the hardware description language (VHDL). This module uses the method of complex multiplication reduction by employing three multiplications, one addition and two subtractions. This is done at the cost of an additional memory table. In the hardware description language program, the twiddle factor multiplier was implemented using component instantiations of three lpm-mult and three lpm-add-sub modules from ALTERA library. Worth to note that lpm modules are supported by most of EDA vendors and LPM provides an architecture-independent library of logic functions or modules that are parameterized to achieve scalability and adaptability [9].

The 16-points FFT processor employing radix-2 algorithm, performs trivial multiplications with $W_{16}^4 = -j$ and W_{16}^0 factors. Multiplication with W_{16}^4 simply can be done by swapping from real to imaginary part and vice versa, followed by changing the sign [7]. The rest of complex multiplications are non-trivial multiplications. However, W_{16}^2 can be implemented with two multiplications and three multiplications for the other non-trivial coefficients. Therefore, the total number of real multiplications in the 16-points FFT scheme is 24, which correspond to 10 complex multiplications.

III. 256-POINTS FFT/IFFT ARCHITECTURAL DESIGN

The radix-2 algorithm is appealing for its simplicity but has the disadvantage for being not adapted for large point FFT calculation such as 256-points FFT, due to the high multiplier requirement. However, to increase the throughput and reduce the power consumption in the 256-points FFT processor, the number of multiplications must be reduced. For this reason, we propose a design methodology for efficient 256-points FFT architecture in order to provide high-speed and keep the area and power consumption as low as possible.

The 256-points FFT/IFFT proposed architecture internally uses two 16-points FFT core for computation. The 256-points FFT with radix-16 decimation of the FFT/IFFT can be formulated in the following way:

$$\text{FFT}(x) = X[k] = \sum_{n=0}^N x[n] \cdot W_N^{nk} \quad (12)$$

We suppose: $N=16T$, $k=s+Tt$, and, $n=l+16m$ where: $s, l \in \{0,1,\dots,15\}$ and $m, t \in \{0,1,\dots,T-1\}$.

We apply this values in equation (12), we obtain:

$$X[s+Tt] = \sum_{l=0}^{15} W_{16}^{lt} [W_{16}^{sl} \sum_{m=0}^{T-1} x[l+16m] \cdot W_T^{sm}] \quad (13)$$

For $T=16, N=256$. The 256-point FFT can be expressed as:

$$X[s+16t] = \sum_{l=0}^{15} [W_{256}^{sl} \sum_{m=0}^{15} x[l+16m] \cdot W_{16}^{sm}] W_{16}^{lt} \quad (14)$$

Equation (14) demonstrates that the implementation of the FFT algorithm for computation of the 256-points FFT (i.e $N=16^2$) involves computation of two 16-points FFTs. This one can be computed by using a radix-16 algorithm as shown in Fig. 2. The first 16-points FFT module computes 16-points of the 256-points FFT on the fitting data slot according to (14) and then multiplies the output with 16×16 inter-dimensional constants coefficients by a multiplier and once again computing the 16-points FFT of the resultant data with the fitting data reordering.

Fig. 2. The 256-points FFT proposed design.

For the 16-points FFT module implementation, the radix-16 algorithm is an attractive algorithm for its requirement of less complex multiplications and additions comparing to radix-4, radix-2 algorithms [8]. However, the use of algorithms with high radix degree increases the complexity of integration in an integrated circuit such as FPGA [14]. Even if the number of nontrivial multiplications and additions/subtractions present a good clue on the effectiveness of an algorithm, hardware integration considers other performance criteria such as the algorithm regularity as well as the conception complexity and architecture control. The gains achieved by the reduction of multiplications or additions could be sometimes lost by the control complexity induced and the interconnection surplus. Therefore, our interest goes to radix-2 algorithm that offers more large regularity for architectural hardware implementation compared to radix-16 and radix-4 and split-radix.

Many communication systems require high throughput and continuous input/output data. The MDC (Multipath Delay Commutator) pipeline architecture is considerably adequate to attain these ends [15], and ideal choice to implement high speed long-size FFT due to its regular structure and simple control [8]. R2MDC (Radix-2 Multipath delay Commutator) structure was adopted as pipeline approach in our design for the purpose of minimizing the memory resources and saving silicon area in FPGA [16]. Moreover, the efficiency of the pipeline FFT processor can be improved by optimizing the structure and saving hardware resources [17].

The proposed architecture of 256-points FFT/IFFT module is illustrated in Fig. 2. Our design consists of an essential unit, the Radix2MDC 16-point FFT unit, which is the kernel of the 256-points FFT processor as interpreted in (14). It has four

stages pipelined structure carrying out 24 real multiplications, and eventually processes 16-points FFT. This block requires an input buffer of size 16 for storing the input serial data in 16 parallel vectors, in order to be arranging for computation by FFT processor according to (14).

The 16-points FFT core uses radix-2 algorithms for computation. The multiplications with $W_{16}^4 = -j$ and W_{16}^0 are trivial, multiplication with W_{16}^4 is simply done by swapping from real to imaginary part and vice versa, followed by changing the sign [18]. The implementation complexity of non-trivial twiddle factors W_{16}^2 and W_{16}^6 is reduced even further, due to replacement of the complex multiplications by basic operations. However, the multiplication with W_{16}^2 and W_{16}^6 was done by add and shift operations [18], [19] (Fig. 3).

Applying permutations, shift-and-add operations with twiddle factors inside the 16-points FFT module instead of complex multiplications reduces the number of expensive multiplication operations. Multiplication with other non-trivial twiddle factors W_{16}^n was implemented with embedded multiplier 9-bit. The employed logic operations allow us to cut down the number of complex multiplications in this optimized approach. Therefore, the total number was reduced to 4 complex multiplications. Consequently, the total number of real multiplication is 12 multiplications [19].

Another approach adopted previously, with which similar design of 256-points FFT implementation was conceived to fulfill the optimum complexity using 16-points FFT module. The total number was reduced to 6 complex multiplications. The total number of real multiplication in this approach is 16 multiplications. The performance comparison of this approach with different FFT pipeline efficient processors is completed in [19].

Fig. 3. Simplification of multiplication complexity inside the 16-points FFT module

IV. PERFORMANCE OF THE PROPOSED ARCHITECTURE

The proposed architecture of 256-points FFT and its core unit, the 16-points FFT computation with the proposed approach was coded in VHDL using Quartus software tool from ALTERA, simulated and synthesized on the low cost ALTERA Cyclone 2 EP2C35F672C6 device. A functional & timing synthesis and simulation were performed.

This 256-points FFT structure achieves different resource reductions; the complex multiplication reduction inside the multiplier unit, combined with the fact of using MDC pipeline architecture. In addition, the adopted simplifications of complex multiplications have reduced multiplication operations inside the 16-points FFT unit.

The 256-points FFT proposed processor requires less number of arithmetic operations compared to the

conventional Cooley-Tukey algorithm, and to efficient processors implemented in pipeline structure with radix-2, radix-2², radix-4 and split-radix algorithm. This comparison is shown in Table I. However, the proposed processor needs more resource usage in terms of Embedded Multiplier (68%) in order to attain low multiplicative complexity. Hence, achieve high speed and low power consumption at the expense of losing logic area.

TABLE I: COMPARISON OF THE PROPOSED ARCHITECTURE OF 256-POINTS FFT WITH DIFFERENT FFT PIPELINE ARCHITECTURES

Complexity	Real Multiplications	Real Add/Sub	Embedded Multiplier 9-bit
Architectures			
Cooley-Tukey	6144	10240	6144
Radix-2MDC	1800	5896	21
Radix-4MDC	1392	5488	27
Split-Radix MDC	1284	5380	33
Radix-2 SDF	1800	5896	21
Radix-4SDF	1392	5896	27
Radix-2 ² SDF	1392	5896	27
Split-Radix SDF	1284	5380	27
Proposed processor	660	5904	48

The comparison below shows that the 256-points FFT proposed in this work requires 10.7%, 36.6%, 47.4% of real multiplications, used respectively in 256-points Radix-2, 256-points radix-4, and 256-points split-radix using Multipath Delay Commutator (MDC) architecture or Single Delay Feedback (SDF) architecture.

For the purpose of verifying the computation accuracy of the 256-points FFT module, we had simulated first, in functional mode the calculation of the unit core of the proposed processor, the 16-points FFT. The output of the implemented FFT core approximately matches the output of an FFT function written in Matlab representing a theoretical example of 16-points FFT calculation. Two same input signals (Square function) were given to Matlab and Quartus tool for simulation [19].

After the computation accuracy of the unit core was verified, we had simulated the calculation of 256-points FFT in functional simulation mode. Two same input signals as rectangular function were given to Matlab and Quartus tool for simulation. The output of simulation in FPGA Cyclone 2 device is shown in Fig. 4. This result is plotted and presented with the output of the FFT obtained in Matlab tool. We succeed to have the implemented 256-points FFT output roughly matches the output of the theoretical model. However, with some noteworthy values differences of the two output signals due to the use of non suitable extension for the FFT fixed point representation.

The 256-points FFT processor was coded in FPGA with hardware description using fixed-point arithmetic while Matlab simulation uses floating point based calculation. The resulting VHDL simulation cannot completely match the one with floating point exactly at the maximum value of the lobes due to the lost of point precision during the computation.

In these simulations, we used 8 bits to represent an FFT point. Using 16 bits, we would have an output signal close to the real one in Matlab. Published results of 16-bits fixed points and 16-bits floating point FFTs in both give highly accurate and comparable results [20]. Consequently, the 16-bits fixed point is more suitable to implement the proposed FFT architecture.

Fig. 4. Simulation of the 256-points FFT processor.

V. CONCLUSION

In this work, the number of multiplications has been considered as a key metric for comparing the FFT performance since it has a large impact on the throughput and power consumption of an FFT processor. The efficient 256-points FFT/IFFT architecture proposed in this paper gives an advantage in terms of multiplicative complexity using pipelined method and complex multiplication reduction approach. The simulation result shows that proposed architecture significantly reduces the number of operations inside the processor compared to other efficient FFT processors. The proposed processor can be integrated with other components to be used as standalone processor applied for OFDM based Wireless Broadband Communication.

REFERENCES

- [1] L. Weidong, "Studies on implementation of lower power FFT processors," *Linköping Studies in Science and Technology*, Linköping, Sweden, 2003.
- [2] A. Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladwy, "Efficient FPGA implementation of FFT/IFFT processor," *Intern. Journal of Circuit, Systems and Signal Processing*, vol. 3, pp.103-110, 2009.
- [3] K. Maharatna, E. Grass, and U. Jagdhold, "A Lower-Power 64-point FFT/IFFT architecture for wireless broadband communication," presented at 7th Inter. Conf. on Mobile Multimedia Communication, Tokyo, 2000.
- [4] W. Li and L. Wanhammar, "Complex multiplication reduction in FFT processor," presented at Swedish System-on-Chip Conference, Falkenberg, Sweden, March, 2002.
- [5] U. M. Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 3rd Ed. Springer, 2007.
- [6] W. H. Chang and T. Nguyen, "An OFDM-specified lossless FFT architecture," *IEEE Transactions on Circuits and Systems*, vol. 53, issue 6, pp. 1235-1243, 2006.
- [7] M. Petrov and M. Glesner, "Optimal FFT architecture selection for OFDM receivers on FPGA," in *Proc. IEEE Intern. Conf. on Field Programmable Technology*, Singapore, 2005, pp. 313-314.
- [8] C. Sahnine, "Architecture of reconfigurable integrated circuit, very high speed and low consumption for digital processing of the advanced OFDM," Ph.D. dissertation, Polytechnic Institute of Grenoble, France, 2009.

- [9] B. Wang, Q. Zhang, T. Ao, and M. Huang, "Design of pipelined FFT processor based on FPGA," in *Proc. 2nd Intern. Conf. on Computer Modeling and Simulation*, Hainan, 2010, pp. 432-435.
- [10] T. Widhe, "Efficient implementation of FFT processing elements," Ph.D. dissertation, Linköping Studies in Science and Technology, Linköping University, Sweden, 1997.
- [11] M. Arioua, S. Belkouch, M. M. Hassani, and M. Agdad, "Complex multiplication reduction in pipeline FFT architecture," in *Proc. of 20th International Conference on Computer Theory and Applications*, Alexandria, Egypt, 2010.
- [12] M. Kannan and S. K. Srivasta, "Low power hardware implementation of high speed FFT core," *Journal of Computer Science*, vol. 3, issue 6, pp. 376-382, 2007.
- [13] S. Simard, J. G. Mailloux, and R. Beguenane, "Optimized FPGA mapping of a bit-serial square root operator with minimum output delay," *International Review on Computers and Software*, vol. 2, issue 6, pp. 661-665, 2007.
- [14] M. A. Jaber, D. Massicotte, and Y. Achouri, "A higher radix FFT FPGA implementation suitable for OFDM systems," in *Proc. The 18th IEEE International Conference on Electronics Circuits and Systems*, Beirut, Dec. 2011, pp. 744-747.
- [15] S. R. Talebiyan and S. Hosseini-Khayat, "Delay analysis of pipeline FFT processors," *International Review on Computers and Software*, vol. 4, issue 3, pp. 422-425, 2009.
- [16] M. Arioua, S. Belkouch, M. M. Hassani, and M. Agdad, "VHDL implementation of an optimized 8-points FFT/IFFT processor in pipeline architecture for OFDM systems," in *Proc. IEEE Intern. Conf. on Multimedia Computing and Systems*, Ouarzazate, Morocco, 2011.
- [17] Z. Dong, Y. M. Zhang, Z. P. Huang, G. L. Tang, and C. W. Liu, "Simulation and application of FFT based pipelined stream," in *Proc. International Conference on Information Engineering and Computer Science*, Wuhan, China, 2009, pp. 1-4.
- [18] Y. Jung, H. Yoon, and J. Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," *IEEE Transactions on Consumer Electronics*, vol. 49, no.1, pp.14-20, 2003.
- [19] M. Arioua, S. Belkouch, and M. M. Hassani, "Efficient 16-points FFT/IFFT architecture for OFDM based wireless broadband communication," *Information Technology Journal*, vol. 11, issue 1, pp. 118-125, 2012.
- [20] M. Adnan, M. M. Al-Akaidi, S. A. Khan, S. Khattak, and M. Assadullah, "Performance evaluation of a hiperlan type 2 standard based on arithmetic formats," *Information Technology Journal*, vol.3, issue 1, pp.1-5, 2004.

M. Arioua received the B.Sc. degree in Telecommunications Engineering and M.S. degree in Telecommunications and Networks from College of Sciences and Technologies of Marrakech in 2005 and 2007, respectively. In 2009, he received M.Sc degree in Multimedia Information Systems, Mobiles and Networks from Picardie Jules Vernes University, Amiens, France. He received his PhD degree in telecommunications & computer science from Cadi Ayyad University, Marrakech in 2012. Currently, he is professor at National School of Applied Sciences in the department of Engineering Technologies, University of Abdelmalek Essaadi Tetouan-Tanger. His research interests include multimedia information systems, networking and communication, wireless communications and mobile computing, real-time processing and embedded systems.

M. M. Hassani received a doctorate thesis in automatic from Nice University, France, in 1982 and Ph. D degree in electrical engineering from Sherbrook University, Canada, in 1992. He is now Professor in the Faculty of Sciences Semailia, Cadi Ayyad University, Marrakech, Morocco. He heads at the University both the Doctoral Studies Centre of Science and Technology and Electronics and Instrumentation Lab. His research interests are in statistical signal processing, nonlinear system, identification and equalization fields.