MIMO Hardware Simulator: Time Domain Versus Frequency Domain Architectures

Bachir Habib, Gheorghe Zaharia, Ghaïs El Zein

Science Journal of Circuits, Systems and Signal Processing

Hardware simulator for MIMO propagation channels: Time domain versus frequency domain architectures

Bachir Habib, Gheorghe Zaharia, Ghais El Zein

Institute of Electronics and Telecommunications of Rennes, IETR, UMR CNRS 6164, Rennes, France

Email adress:
Bachir.habib@insa-rennes.fr (B. Habib)

Abstract: A hardware simulator facilitates the test and validation cycles by replicating channel artifacts in a controllable and repeatable laboratory environment. This paper presents an overview of the digital block architectures of Multiple-Input Multiple-Output (MIMO) hardware simulators. First, the simple frequency architecture is presented and analyzed. Then, an improved frequency architecture, which works for streaming mode input signals, is considered. After, the time domain architecture is described and analyzed. The architectures of the digital block are presented and designed on a Xilinx Virtex-IV Field Programmable Gate Array (FPGA). Their accuracy, occupation on the FPGA and latencies are analyzed using Wireless Local Area Networks (WLAN) 802.11ac and Long Term Evolution System (LTE) signals. The frequency and the time approaches are compared and discussed, for indoor (using TGn channel models) and outdoor (using 3GPP-LTE channel models) environments. It is shown that the time domain architecture present the best solution for the design of the architecture of the hardware simulator digital block. Finally, a 2×2 MIMO time domain architecture is described and simulated with input signal that respects the bandwidth of the considered standards.

Keywords: Hardware simulator; MIMO radio channel; FPGA; 802.11ac; LTE.

1. Introduction

The need to improve the performance of wireless networks has led to an increased interest in MIMO communication techniques which offer high data bit rates for wireless systems. The current communication standards show a clear trend in industry to support MIMO functionality. Several studies published recently present systems that reach a MIMO order of 8×8 and higher [1]. This is made possible by advances at all levels of the communication platform as, for example, the monolithic integration of antennas [2] and the design of the simulator platforms [3].

To evaluate the performance of the recent communication systems, a channel hardware simulator is considered using the recent communication standards based on MIMO techniques. It provides the processing speed required to the evaluation of performance in real-time and allows comparing various systems in the same test conditions. These simulators are standalone units that provide the fading signals in the form of analog or digital samples [4, 5].

With the continuous increase of FPGA capacity, entire baseband systems can be efficiently mapped onto faster FPGAs for more efficient prototyping, testing and verification. As shown in [6], the FPGAs provide the greatest flexibility in algorithm design and visibility of resource utilization. Also, they are ideal for rapid prototyping and research use such as testbed [7].

The simulator is reconfigurable with standards bandwidth not exceeding 100 MHz, which is the maximum for FPGA Virtex-IV. However, in order to exceed 100 MHz bandwidth, more performing FPGA as Virtex-VI can be used [3]. The simulator is configured with LTE and WLAN 802.11ac standards. The channel models used by the simulator can be obtained from standard channel models, as the TGn 802.11n channel models [8] and 3GPP-LTE channel models [9], or from real measurements conducted with the MIMO channel sounder designed and realized at IETR [10-12].

At IETR, several architectures of the digital block of a hardware simulator have been studied, in both time and frequency domains [12, 13]. Moreover, [14] presents a new method based on determining the parameters of a channel simulator by fitting the space time-frequency cross-correlation matrix of the simulation model to the estimated matrix of a real-world channel. This solution shows that the error obtained can be important.

Typically, wireless channels are commonly simulated using Finite Impulse Response (FIR) filters, as in [13, 15, 16, 17]. Moreover, the Fast Fourier Transform (FFT) module can be used to obtain an algebraic product. Thus, frequency architectures are presented, as in [13, 15].

The contributions and the structure of this paper are organized as follows.

Section 2 presents the channel models and the Kronecker

method used to obtain time-varing channel.

In Section 3, the simple frequency architecture is studied. Then, it is implemented on an FPGA Virtex-IV from Xilinx. The occupation on the FPGA of the architecture, the accuracy of the output signals and its latency are given. The second part of this section presents an improved frequency architecture that accepts long input signals [18]. In fact, the simple frequency architecture limits the input signals to the size of the FFT/IFFT blocks. Moreover, if the signal is larger than the size of the FFT/IFFT blocks, tests will show that if we split the input signals to parts equal to the size of the FFT/IFFT blocks, it will present an error at the output. Therefore, in this section the improved frequency architecture is analyzed, tested and verified.

Section 4 presents a description of the time domain architecture. Then, it is implemented on an FPGA Virtex-IV. The occupation on the FPGA of the architecture, the accuracy of the output signals and its latency are given.

In Section 5, after comparing the improved frequency domain architecture and the time domain architecture, we have chosen the time domain architecture which has a better occupation on the FPGA, better latency and better precision.

For now, the comparison of the previous architectures was made using a SISO channel and long input signals to show their validation in the worst conditions. However, after choosing the best architecture, more realistic conditions have to be considered. Therefore, in Section 6, tests are made with input signal that respects the bandwidth chosen between $[\Delta, B + \Delta]$ and by considering 2×2 MIMO architecture. In fact, the channel impulse responses can be presented in baseband with its complex values, or as real signals with limited bandwidth B between $f_c - B/2$ and $f_c + B/2$, where f_c is the carrier frequency. In this paper, to eliminate the complex multiplication and the f_c, the hardware simulation operates between Δ and $B + \Delta$, where Δ depends on the band-pass filters (RF and IF). The value Δ is introduced to prevent spectrum aliasing. In addition, the use of a real impulse response allows the reduction by 50% of the size of the FIR filters and by 4 the number of multipliers. Thus, within the same FPGA, larger MIMO channels can be simulated.

Lastly, Section 7 gives concluding remarks and prospects.

2. Channel Model

A MIMO propagation channel is composed of several time variant correlated SISO channels. For MIMO 2×2 channel, the received signals $y_j(t,\tau)$ can be calculated using a convolution :

(1)

The associated spectrum is calculated by the Fourier transform (using FFT modules):

(2)

The development of the digital block of a channel hardware simulator requires a good knowledge of the propagation channel. The different models of channels presented in literature used to apprehend as faithfully as possible the behavior of the channel.

Two channel models are considered to cover indoor and outdoor environments: the TGn channel models (indoor) and the 3GPP-LTE channel models (outdoor). Moreover, using the channel sounder realized at IETR, measured impulse responses are obtained for specific environments: shipboard, outdoor-to-indoor.

2.1. TGn Channel Models

TGn channel models [8] have a set of 6 profiles, labeled A to F, which cover all the scenarios. Each model has a number of clusters. For example, model E has four clusters. Each cluster corresponds to specific tap delays, which overlaps each other in certain cases. Reference [8] summaries the relative power of the impulse responses for TGn channel model E by taking the Line-Of-Sight (LOS) impulse response as reference. According to the standard and the bandwidth, the sampling frequency is f_s = 165 MHz and the sampling period is $T_s = 1/f_s$.

2.2. 3GPP-LTE Channel Models

3GPP-LTE channel models are used for mobile wireless applications. A set of 3 channel models is used to simulate the multipath fading propagation conditions. A detailed description is presented in [9]. For LTE signals, f_s = 50 MHz.

2.3. Time-Varying Channels

In this section, we present the method used to obtain a model of a time variant channel, using Rayleigh fading [19] and based on Kronecker model [20].

The Doppler frequency f_d is equal to:

— (3)

where c is the celerity and v is the environmental speed. We have chosen a refresh frequency $f_{ref} > 2.f_d$ to respect the Nyquist-Shannon sampling theorem.

For an indoor environment (TGn model E for example), at f_c = 5 GHz and v = 4 km/h, f_d = 18.51 Hz. Thus, we have chosen a refresh frequency f_{ref} = 40 Hz. For an outdoor environment (3GPP-LTE model EVA for example), at f_c = 1.8 GHz and v = 80 km/h, f_d = 133.27 Hz. Thus, we have chosen a refresh frequency f_{ref} = 300 Hz.

The MIMO channel matrix H can be characterized by two parameters:

1) The relative power P_c of constant channel components which corresponds to the LOS.

2) The relative power P_s of the channel scattering components which corresponds to the Non-Line-Of-Sight (NLOS).

The ratio P_c/P_s is called Ricean K-factor.

Assuming that all the elements of the MIMO channel matrix H are Rice distributed, it can be expressed for each tap by:

(4)

where H_F and H_V are the constant and the scattered channel matrices respectively.

The total relative received power $P = P_c + P_s$. Therefore:

(5)

(6)

If we combine (5) and (6) in (4) we obtain:

(7)

To obtain a Rayleigh fading channel, K is equal to zero, so H can be written as:

(8)

P is derived from [8] or [9] for each tap of the considered impulse response. For 2 transmit and 2 receive antennas:

(9)

where X_{ij} (i-th receiving and j-th transmitting antenna) are correlated zero-mean, unit variance, complex Gaussian random variables as coefficients of the variable NLOS (Rayleigh) matrix H_V.

To obtain correlated X_{ij} elements, a product-based model is used [20]. This model assumes that the correlation coefficients are independently derived at each end of the link:

(10)

H_w is a matrix of independent zero means, unit variance, complex Gaussian random variables. R_r and R_t are the receive and transmit correlation matrices. They can be written by:

(11)

where is the correlation between channels at two receives antennas, but originating from the same transmit antenna (SIMO). In other words, it is the correlation between the received power of channels that have the same Angle of Departure (AoD). is the correlation coefficient between channels at two transmit antennas that have the same receive antenna (MISO).

The use of this model has two conditions:

1) The correlations between channels at two receive (resp. transmit) antennas are independent from the *Rx* (resp. *Tx*) antenna.
2) If s_1, s_2 are the cross-correlation between antennas of the same side of the link, then :
   - $s_1 =$ + .
   - $s_2 =$ + .

For the uniform linear array, the complex correlation coefficients and are expressed by :

(12)

where $D = 2\pi d/\lambda$, $d = 0.5\lambda$ is the distance between two successive antennas, λ is the wavelength and R_{xx} and R_{xy} are the real and imaginary parts of the cross-correlation function of the considered correlated angles:

(13)

(14)

The Power Angular Spectrum (PAS) closely matchs the Laplacian distribution [21, 22]:

(15)

where σ is the standard deviation of the PAS (which corresponds to the numerical value of AS).

3. Frequency Domain Architecture Design

3.1. Simple Frequency Domain Architecture

3.1.1. Description

In the frequency domain, the architecture for the digital part of the hardware simulator for a SISO channel can be represented by Fig. 1 which describes the digital representation of signals. This architecture uses a Xilinx module performing the FFT and which can be configured to perform as IFFT. The complex multiplier, the memory block and the truncation module will also be detailed.

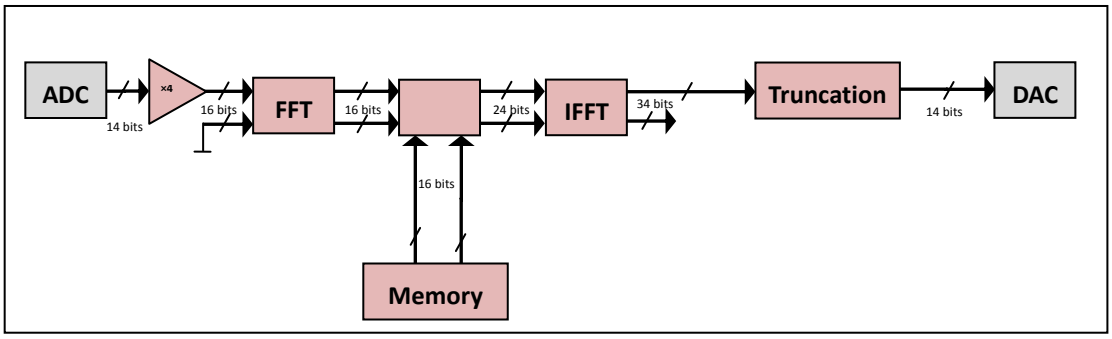


Figure 1. *Diagram of a SISO channel in simple frequency domain with digital representation.*

The memory block is used to store the frequency response profiles of the considered channel. The real and imaginary parts of the frequency response are quantified on 16 bits to have a satisfied precision. There are two methods to load the frequency response to the FPGA. The first by saving the frequency responses on the RAM block of the Virtex-IV. By this method, the transfer is made just one time before the compilation of the VHDL program. However, we the number of the RAM blocks and their size are limited, especially for time variant channels to simulate a lot of profiles. In a Virtex-IV, there is 192 RAM blocks of 18 kbit each. For a 2×2 MIMO channel with N_F = 512 for outdoor environment (N_F is the size of the FFT/IFFT block), then there are 4 SISO channels to simulate, Thus, four frequency response profiles are needed. The data send is equal to 512 × 4 samples of 32 bits, or 2048 samples of 32 bits. Therefore, 65.536 kbits to transmit for a profile. The number of frequency profiles that can be saved in the RAM blocks of the Virtex-IV is 192×18/65.536 = 52 profiles. Thus, 13 profiles for each SISO channel. For N_F = 32 (in indoor), 210 profiles for each SISO channel. If these profile numbers are sufficient for the test, we can add a function in the VHDL program which is used to load the profiles by running the address of the RAM blocks in a sinusoidal manner. To load a large number of profiles, the second method consist on using a bus transfer between the computer and the RAM block in the FPGA. The profiles containing these 32 bits samples are stored in a text file on the hard disk of a computer. Then, This file loads the memory block which will supply the hardware simulator. The transfer can be done either by the USB 1.1 interface, either by the PCI interface, both available on the prototyping board used.

In the worst case, which is for 3GPP-LTE model ETU, f_{ref} =150 Hz. The refresh period T_{ref} =1/ f_{ref} during which we must refresh all 4 profiles, which will make a rate of:

$$\frac{8\times 192}{6666\times 10^{-6}} = 1.22 \quad MBps \tag{16}$$

The USB bus does not meet this rate. Thus, the PCI bus has been selected to load the frequency response profiles. It has a rate up to 30 MBps. In addition, the PCI bus is a 32 bit bus, so on every clock cycle, it transmits a complex sample of the frequency response. Moreover, as a SISO channel here corresponds to a profile (512 × 32) bits, or 2048 bytes, the rate of 30 MBps allows us to load 97 SISO channels during the refresh time T_{ref}.

The block diagram in Fig. 2 shows the connection between the PC that contains the file of the frequency response profiles and the card XtremeDSP of Nallatech containing the Virtex-IV where its digital block is shown.

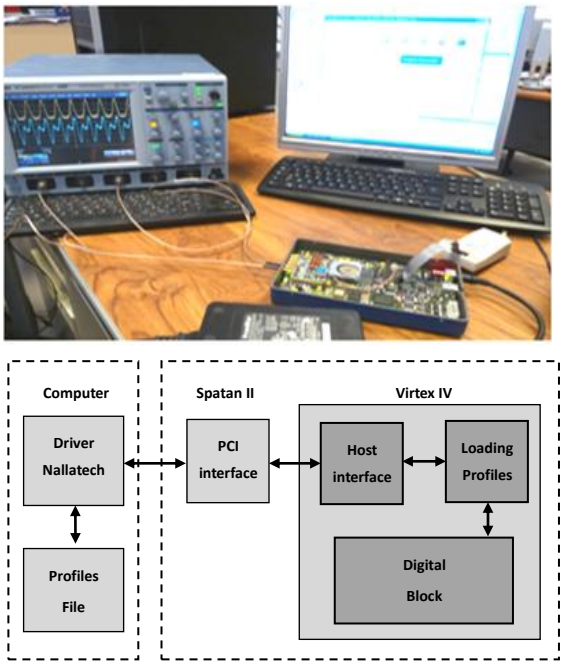


Figure 2. *Connection between the computer and the XtremeDSP board.*

The programmable component Spartan-II is especially dedicated to the treatment of the USB and PCI interfaces. It has been programmed by Nallatech to collect data on the bus and redirect them to the Virtex-IV where our architecture is implemented.

An IP called "Host Interface" reads the data from the PCI bus and store them in FIFO memory. Then the module called "Loading profiles" reads and distributes the values of samples in the two blocks RAM or double port memory block, called "RAM_A" and "RAM_B" as we can see from the following Fig. 3. This figure details the connection between the IP "Host Interface" and the loading profile block.

The two blocks RAM are used to read a profile while loading another. In fact, a signal *S* control in one hand the demultiplexer, and on the other hand controls the multi-

plexer. Thus, when the multiplexer selects a RAM block to read the 32 values of a complex frequency response profile, the demultiplexer selects another RAM block to write the 32 values of the following profile. Thus, while a profile is used, the following profile is loaded and will used after T_{ref}. The signal S is periodic with a period equal $2.T_{ref}$. This method is based on a double buffer operation. The output of the multiplexers is a 32 bit bus with 16 MSB directed to the input of the real samples and 16 LSB to the input of the imaginary samples of the complex multiplier.

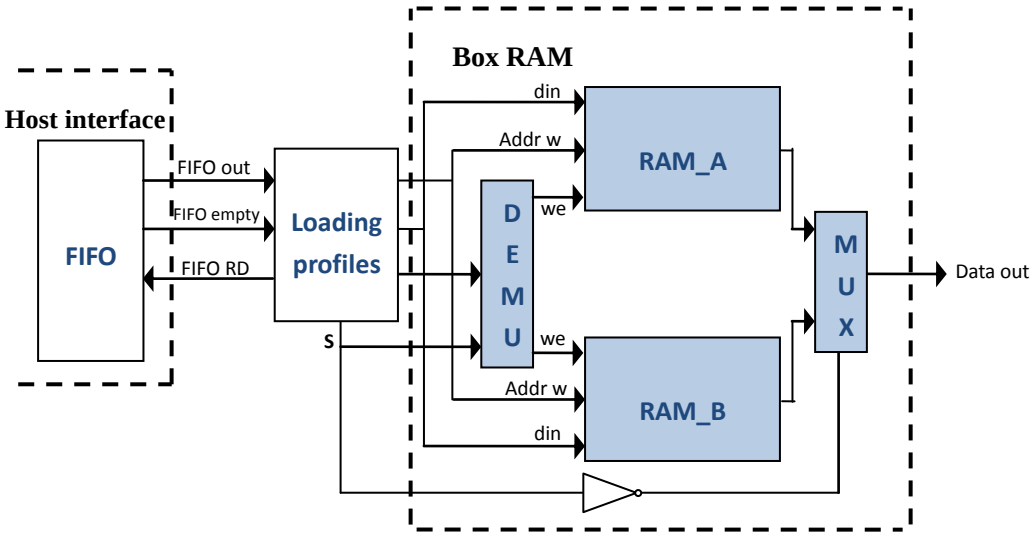


Figure 3. *Loading of the frequency response.*

The complex multiplier uses the "XtremeDSP" presented on the FPGA, which contains a multiplier of 18 ×18 bits, an adder of 48 bits and a register. After the multiplication, the length of the samples can be up to 128 bits. These multipliers have an internal truncation to provide the user the needed number of bits at the output.

The calculated values of the output of the IFFT block are quantified on M_y = 34 bits. The truncation block, located after the IFFT Xilinx block, is necessary to reduce the number of bits of the output samples of the IFFT block to n_{DAC} = 14 bits so these samples can be accepted by the DAC, while keeping the best possible accuracy. Unlike blocks presented above, this block has been programmed. The easiest immediate solution is to keep the 14 MSB. However, for low values , keeping only the MSB can cause null values at the input of the DAC while they were non-null at the output of the IFFT block.

Therefore, instead of a simple brutal truncation, which keeps the first 14 bits starting with the MSB, we considered a sliding window truncation of 14 bits. This truncation is illustrated in Fig. 4 and it considers the most significant bits.

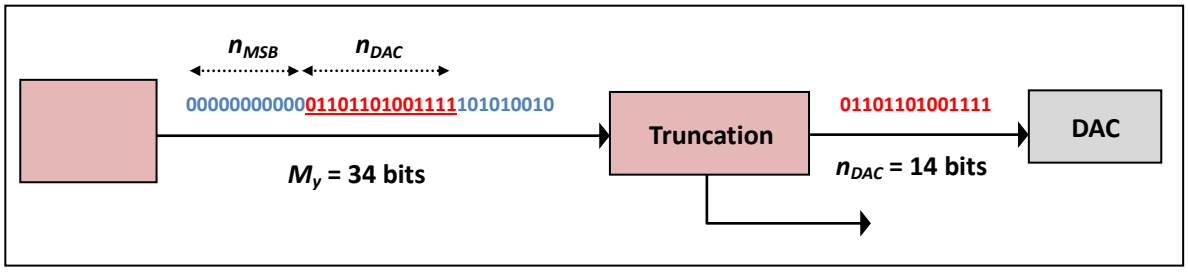


Figure 4. *Sliding window truncation.*

This truncation modifies each output sample. Therefore, a reconfigurable amplifier after the DAC must be used to restore the correct output value by multiplying it by a scale factor of .

3.1.2. Implementation results

In this section, the implement result of the simple frequency architecture on the FPGA is presented. First, we describe the choice of the input signal used for the test. Then, we implement the architecture on the Virtex-IV which consists the digital block of the hardware simulator.

In this Section, the simple frequency architecture is tested with WLAN 802.11ac and LTE signals for different environments. We have chosen to simulate the TGn model E for WLAN 802.11ac signals and 3GPP-LTE model EVA for LTE signals because they need the same size N_F = 128 of FFT/IFFT modules. In that way, a comparison of the architecture in indoor and outdoor can be made.

Fig. 5 presents the vector H of TGn model E where with $W_t = 128.T_s$. For WLAN 802.11ac, T_s is the sampling period and it is equal to 1/(165MHz).

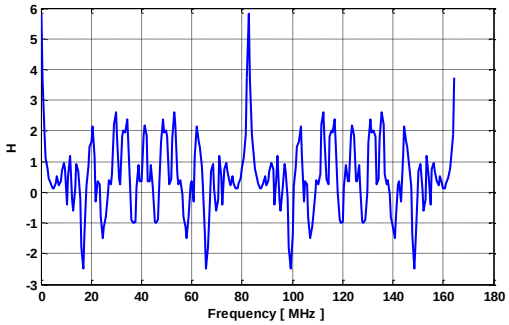


Figure 5. *Frequency response of TGn model E.*

Fig. 6 presents the vector H of 3GPP-LTE model EVA. For LTE, T_s =1/(50MHz).

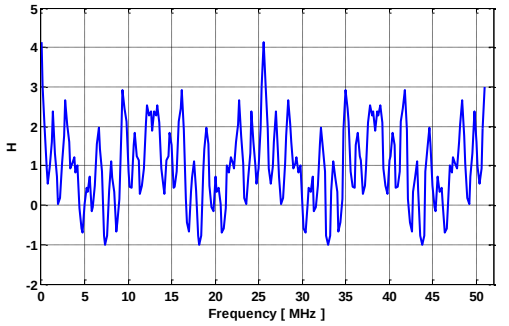


Figure 6. *Frequency response of 3GPP-LTE model EVA.*

The H vector is implemented and saved on a RAM block in the FPGA Virtex-IV.

A Gaussian input signal $x(t)$ is considered. In fact, the use of a Gaussian signal is preferred because it has a limited duration in both time and frequency domains. Thus, its Fourier Transform can be calculated by FFT block of limited size. The $x(t)$ size is limited by the size of the FFT/IFFT module used in the simple frequency architecture. The $x(t)$ used to test the simple frequency architecture is computed by:

(17)

The center of each Gaussian and each are chosen in a way to show the effect of each path of the taps of the impulse responses on the output signal. The parameters depend on the channel and the standard used. The WLAN 802.11ac signals uses a sampling frequency f_s = 165 MHz and a sampling period T_s =1/ f_s . The last Excess Time Delay (ETD) for TGn model E is 730 ns. Therefore, the size of the FFT/IFFT module will be equal to $730/T_s = 120$ and rounded to N_F =128 (to be written in the form of 2^n where n is an integer). Thus, $W_t = N_F.T_s$, $m_{x1} = W_t/8$, $m_{x2} = 3.W_t/8$, $\sigma_1 = m_{x1}/8$ and $\sigma_2 = m_{x2}/20$. This input signal named $x_{WLAN}(t)$ is presented in Fig. 7.

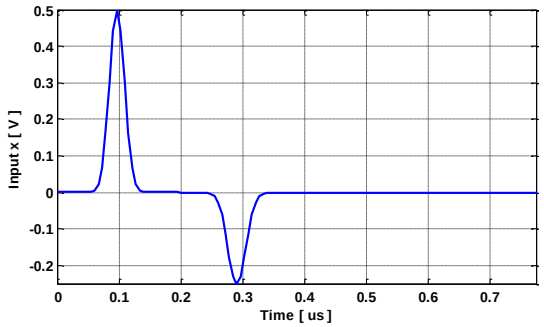


Figure 7. *Input signal for WLAN 802.11ac for the simple frequency architecture test.*

The ADC and DAC of the development board have a full scale [$-V_m$,V_m], with V_m = 1 V. For the simulations we consider $x_{m1} = V_m/2$ and $x_{m2} = -V_m/4$.

To compare later the results, it is better to use the same input signal that covers the same area of W_t. Thus, we will use the same signal but with LTE parameters. In that way, only the scale factor of the time axis changes. For LTE signals, f_s = 50 MHz. The last ETD for 3GPP-LTE model EVA is 2510 ns. Therefore, the size of the FFT/IFFT module will be equal to $2510/T_s$ = 125 and rounded to N_F =128. Thus, $W_t = N_F.T_s$, $m_{x1} = W_t/8$, $m_{x2} = 3.W_t/8$, $\sigma_1 = m_{x1}/8$ and $\sigma_2 = m_{x2}/20$. This input signal named $x_{LTE}(t)$ is presented in Fig. 8.

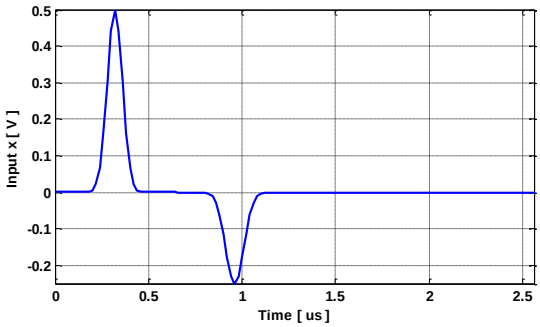


Figure 8. *Input signal for LTE for the simple frequency architecture test.*

The occupation on the FPGA is obtained after performing three main operations from the program written in VHDL: the synthesis, the mapping and the place and route. The synthesis is the compilation of a functional description of a circuit to generate a diagram with logic gates and flip-flops. Then the mapping operation describes the combination of these logic gates as LUT, which is a kind of correspondence table as static memory, which allows combining pre-computed values. Finally, after component placement, the routing provides the connection arrangements between logic resources and I/O hardware component.

Table 1 shows the device utilization in one Virtex-IV

SX35 for one SISO channel using the simple frequency architecture for the TGn model E.

Table 1. *Virtex-IV utilization for SISO simple frequency domain architecture for TGn model E*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	3,133	30,720	11%
Number of LUT	3,844	30,720	13%
Number of occupied slices	1,789	15,360	12%
Number of DSP block	20	192	11%
Number of RAM block	9	192	5%

Table 2 shows the device utilization in one Virtex-IV SX35 for one SISO channel using the simple frequency architecture for the 3GPP-LTE model EVA.

Table 2. *Virtex-IV utilization for SISO simple frequency domain architecture for 3GPP-LTE model EVA*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	3,472	30,720	12%
Number of LUT	4,292	30,720	14%
Number of occupied slices	1,992	15,360	13%
Number of DSP block	20	192	11%
Number of RAM block	9	192	5%

The two channels use FFT/IFFT module of size 128. Therefore, there occupation on the FPGA is almost the same. For TGn model E the sampling frequency is higher than that of 3GPP-LTE model EVA. Thus, it uses more LUT blocks. The FFT block that has a 16 bits input and 16 bits output, needs 3 DSP blocks and 3 RAM block. The IFFT block that has a 24 bits input and 34 bits output, needs 14 DSP blocks and 5 RAM block. Moreover, 3 DSP block are added which are used by the complex multiplier, and 1 RAM block is added to save the channel frequency response.

With a SISO channel, the slice occupation is between 12 and 13, thus a 2×2 MIMO channel can easily be implemented with the additional MIMO circuit.

3.2. Improved Frequency Domain Architecture

3.2.1. Description

The simple frequency architecture limits the input signal to the size of the FFT/IFFT blocks. Moreover, if the signal is larger than the size of the FFT/IFFT blocks, tests will show that if the input signal is split to parts equal to the size of the FFT/IFFT blocks, it will present an error at the output. Therefore, an improved frequency architecture is proposed.

To test the architecture with modeled impulse responses, the output can't be predicted. Thus, we present firstly the parameters used for the test of the simple frequency architecture. Secondly, the cause of using a new improved architecture will be presented. Finally, the new improved frequency architecture will be introduced and analyzed.

To test the architecture with an input signal in streaming mode, we use test signals, simple to treat and with a possible prediction of their output signal. In fact, the results obtained with these test signals must be obtained by theoretical calculation. Thus, the ideal case is to use an input signal for the test with finite window in both the time domain and in the frequency domain. The Gaussian signal meets these criteria. The Gaussian is a good trade off for a finite number of points in both frequency and time domains. Thus, to test the architecture, we will use in one hand a Gaussian that stands for by input signal *x(t)*, and on the other hand, a Gaussian signal for the impulse response *h(t)*.

In the frequency domain which interests us here, we will use the Gaussian *H(f)*, which is the FT of the Gaussian *h(t)*, to represent the frequency response that will feed the simulator. The output *y(t)* will also be a Gaussian.

As we shall obtain the output signal *y(t)* given by the relation:

$$y(t) = x(t) * h(t) \tag{18}$$

We express the signals *x(t)*, *h(t)* and *y(t)* by:

$$x(t) = x_m e^{-\frac{(t-m_x)^2}{2\sigma_x^2}} \tag{19}$$

$$h(t) = h_m e^{-\frac{(t-m_h)^2}{2\sigma_h^2}} \tag{20}$$

$$y(t) = y_m e^{-\frac{(t-m_y)^2}{2\sigma_y^2}} \tag{21}$$

Their FFT gives the following Gaussian frequency signals:

$$X(f) = x_m \sigma_x \sqrt{2\pi} e^{-2\pi^2 f^2 \sigma_x^2} e^{-j2\pi f m_x} \tag{22}$$

$$H(f) = h_m \sigma_h \sqrt{2\pi} e^{-2\pi^2 f^2 \sigma_h^2} e^{-j2\pi f m_h} \tag{23}$$

$$Y(f) = y_m \sigma_y \sqrt{2\pi} e^{-2\pi^2 f^2 \sigma_y^2} e^{-j2\pi f m_y} \tag{24}$$

As the convolution in the time domain can be replaced by the multiplication in the frequency domain, we obtain:

$$Y(f) = X(f).H(f) \tag{25}$$

$$= x_m h_m \sigma_x \sigma_h \left(\sqrt{2\pi}\right)^2 e^{-2\pi^2 f^2 (\sigma_x^2+\sigma_h^2)} e^{-j2\pi f (m_x+m_h)} \tag{26}$$

Identifying the elements of , we obtain:

$$m_y = m_x + m_h \tag{27}$$

$$\sigma_y^2 = \sigma_x^2 + \sigma_h^2 \tag{28}$$

$$y_m \sigma_y \sqrt{2\pi} = x_m h_m \sigma_x \sigma_h \left(\sqrt{2\pi}\right) \tag{29}$$

The value of the amplitude of the Gaussian *x(t)* and *y(t)* must be selected in order to obtain good precision of the digital signal. The ADC/DAC have a full scale of [$-V_m$, V_m], with V_m=1V. Thus, we have chosen $x_m = y_m = V_m/2$ to avoid the clipping problems and because the output signal *y(t)* must not leave the range of [$-V_m$, V_m]. h_m becomes:

$$h_m = \frac{\sigma_y}{\sqrt{2\pi}\sigma_x \sigma_h} \tag{30}$$

The test will be made with T_s=20 ns which is used by LTE. Moreover, to test the simple frequency architecture with a streaming input signal, we have chosen the size of the FFT/IFFT blocks of N_F = 128 (the same as the previous Section) and the window of the input signal equal to $3W_t = 3N_F$. The other parameters of the input Gaussians are determined by: $m_x = W_t/2$ and = $m_x/2$. For *H(f)*, its window is equal to W_t, $m_h = W_t/2$ and = $m_h/2$.

The samples of the quantified Gaussian input *x(t)*, put in the VHDL program and generated by MATLAB, are used as input of the FFT block. The quantified Gaussian frequency *H(f)* is stored in a RAM block.

The FFT 512 block will split the corresponding quantized input vector *x* in three sub input signals (x_1, x_2 and x_3) of N_F = 128 samples each.

Applying these parts to the input of simple frequency architecture whose frequency response is *H*, we obtain three sub-output vectors y_1, y_2 and y_3. To validate the streaming mode, a comparison is made between the concatenation of these three vectors and the theoretical signal *y(t)* obtained by a convolution, as shown in Fig. 9.

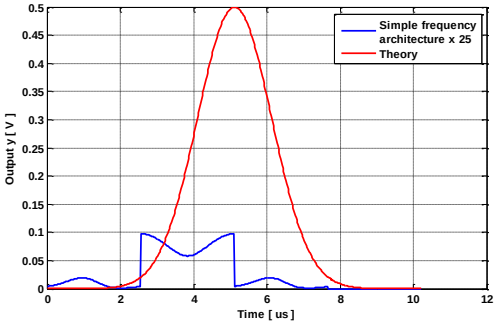


Figure 9. *The simple frequency output versus the theory output for input in streaming mode.*

The concatenation of the three sub-outputs obtained by the simple frequency architecture gives a wrong result if we compare it to the theory output signal. As we notice, the output signal using the simple frequency architecture is obtained on a window equal to $3N_FT_s$ = 3×512×20ns = 30.72 s. However, the correct result is obtained on $4N_FT_s$ = 4×512×20ns = 40.96 s.

In fact, each partial result y_1, y_2 and y_3 must have $2N$ samples equal in time to $2\times512\times T_s$ = 20.48 s (if x_1, x_2, x_3 and h have N_F samples). Using the simple frequency architecture, the IFFT block gives its result only with N_F samples. There is a truncation of each partial result y_i. Thus, the concatenation of these partial results gives a wrong result.

Therefore, an improved frequency architecture is proposed as a solution. It is presented in Fig. 10 and it operates using two FFT/IFFT blocks of 256 points.

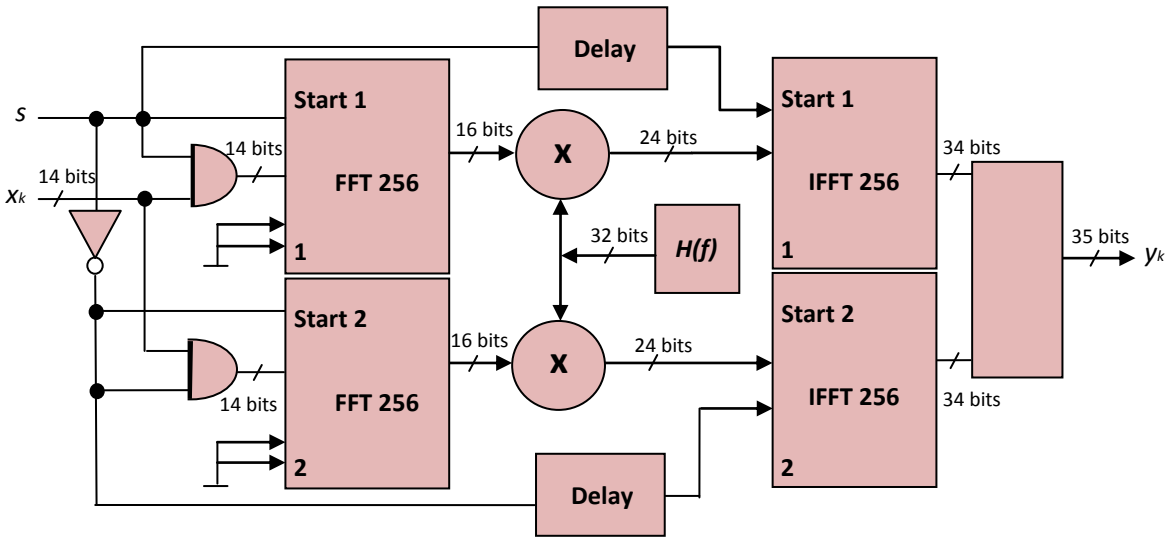


Figure 10. *Improved frequency domain architecture for one SISO channel.*

This solution consists on completing each vector x_i with N_F zeros and on using the FFT/IFFT blocks with size two times larger ($2N_F$). Each FFT module operates with 16 bit input samples, and has a 12 bit phase factor. The switch signal S provides alternated use of the FFT modules. The start input of the FFT modules is active on the rising edge of the switch signal S.

Fig.11 presents the theory output signal versus the output signal obtained by using the improved frequency domain architecture.

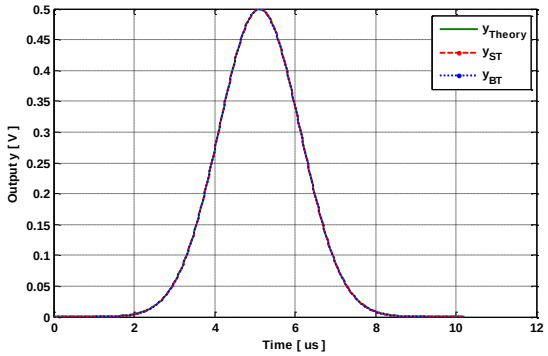
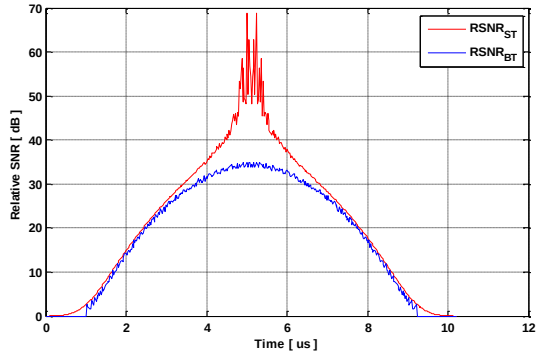


Figure 11. *Output signal using the improved frequency domain architecture versus the theory.*

Fig. 12 presents the relative error and the relative SNR (computed by the formulas in Section 5) of the output signal using the improved frequency domain architecture, versus the theory output signal, using Brutal (BT) or Sliding window Truncations (ST).



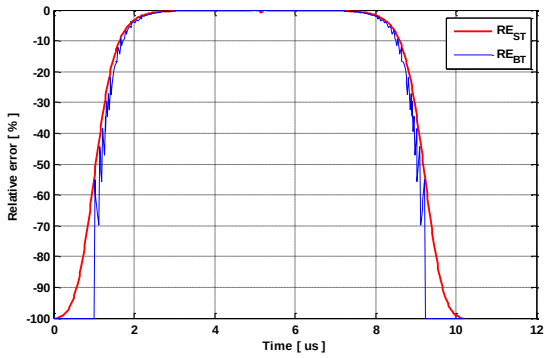


Figure 12. *Relative error and relative SNR using the improved frequency domain architecture.*

3.2.2. Implementation on FPGA

In this section, we will use an input Gaussian signal $x(t)$ large enough to test the improved frequency architecture in streaming mode. For TGn model E and 3GPP-LTE model EVA, $N_F = 128$ as we have seen previously. The window of the input signal $x(t)$ is chosen equal to $W_t = 3N_FT_s = 384T_s > N_FT_s$.

The input signal $x_{WLAN}(t)$ is presented in Fig. 13.

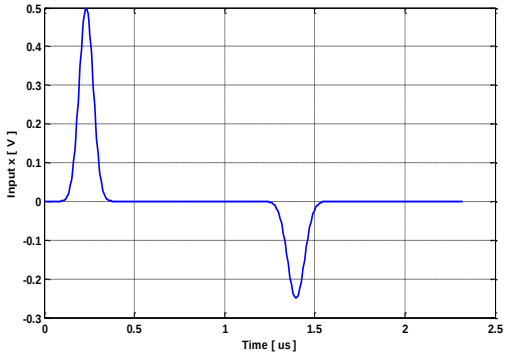


Figure 13. *Input signal for WLAN 802.11ac for the improved frequency architecture.*

The input signal $x_{LTE}(t)$ is presented in Fig. 14.

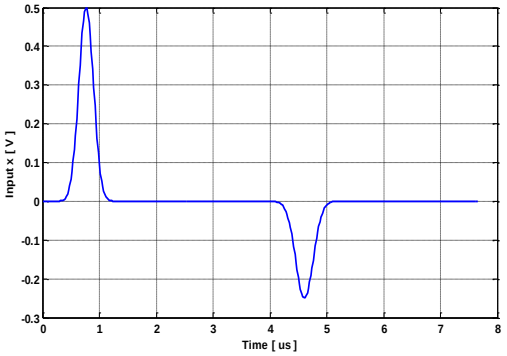


Figure 14. *Input signal for LTE for the improved frequency architecture.*

Table 3 shows the device utilization in one Virtex-IV SX35 for one SISO channel using the improved frequency architecture for the TGn channel model E.

Table 3. *Virtex-IV utilization for SISO improved frequency domain architecture for TGn model E*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	6,902	30,720	23%
Number of LUT	8,033	30,720	27%
Number of occupied slices	3,843	15,360	26%
Number of DSP block	40	192	21%
Number of RAM block	19	192	10%

Table 4 shows the device utilization in one Virtex-IV SX35 for one SISO channel using the simple frequency architecture for the 3GPP-LTE model EVA.

Table 4. *Virtex-IV utilization for SISO improved frequency domain architecture for 3GPP-LTE model EVA*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	7,265	30,720	24%
Number of LUT	8,365	30,720	28%
Number of occupied slices	4,114	15,360	27%
Number of DSP block	40	192	21%
Number of RAM block	18	192	10%

The improved frequency architecture using FFT/IFFT modules of size 256 occupy between 26 to 27 % of slices on the FPGA for one SISO channel. Thus, it has very high occupation. Therefore, it is impossible to implement a 2×2 MIMO system using this architecture on an FPGA Virtex-IV. However, a 2×1 MIMO system or a 1×2 MIMO system can be implemented.

In the case of impulse responses that have a large excess delay, but a small number of non-null taps, a large size for the FFT/IFFT modules is needed. Therefore, the occupation increases significantly. In this case, the time domain architecture can be used which is analyzed in details in the Section.

4. Time Domain Architecture Design

4.1. Description

The block diagram of the digital architecture of the hardware simulator in the time domain is shown in Fig. 15 for one SISO channel.

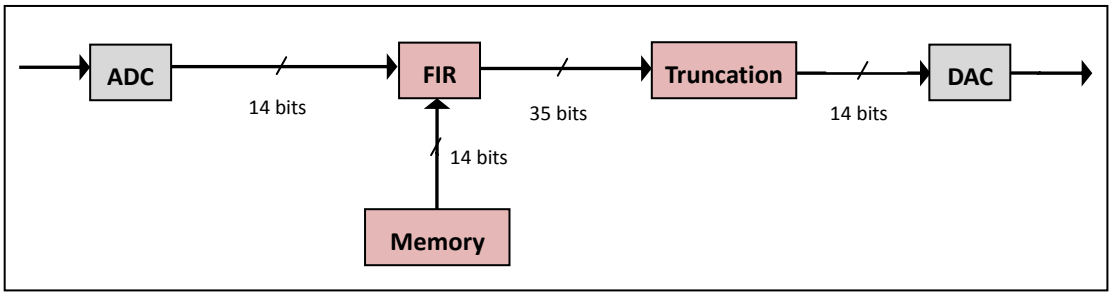


Figure 15. *Diagram of a SISO channel in time domain with digital representation.*

The time domain approach is based on a convolution between the input signal *x(t)* and the channel impulse response *h(t)*.

This convolution product can be presented, as in Fig. 16, which shows a FIR *N* filter architecture, with 18 multipliers, for one SISO channel.

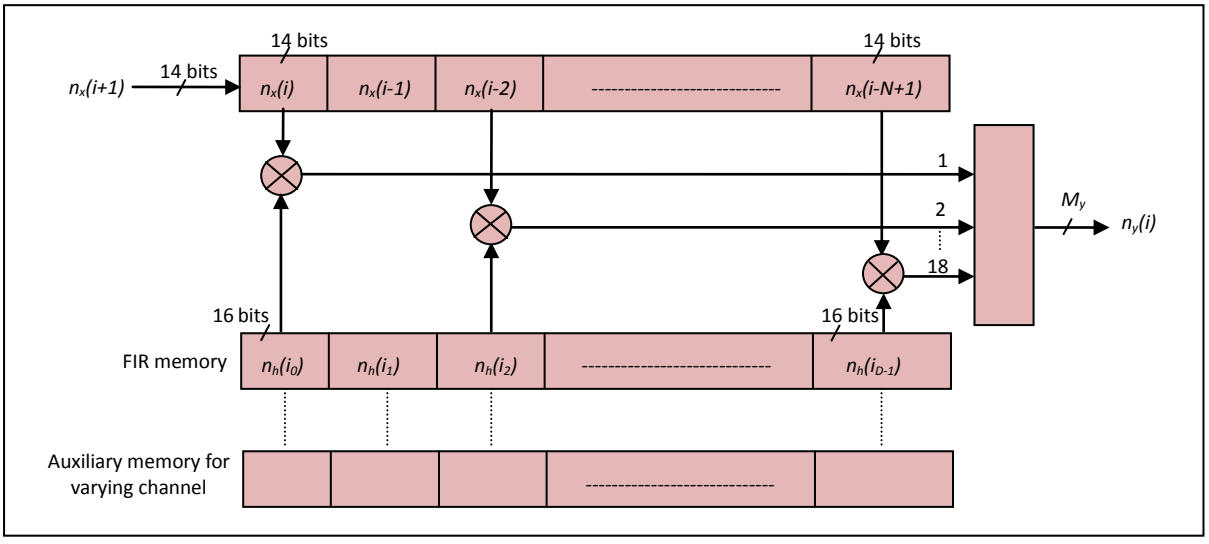


Figure 17. *Time domain architecture for one SISO channel, for 3GPP-LTE model ETU.*

The number of bits at the output before the truncation is equal to:

$$M_y = M_x + M_h + M_{tap} \tag{31}$$

where M_x = 14 bits is the number of bits of the input signal, M_h = 16 is the number of bits of the impulse response and M_{tap} can be expressed by:

$$M_{tap} = \lceil \log_2(D) \rceil \tag{32}$$

For 3GPP-LTE channel model ETU, N = 250 and the

number of taps equal to 9. Thus, M_y = 34 bits. For TGn channel model E, N = 121 and the number of taps equal to 18. Thus, M_y = 35 bits.

The profiles contain 24 bits samples (16 bits for the relative power of the impulse responses and 8 bits for their excess delays) and they are stored in a text file on the hard disk of a computer. Then, This file loads the memory block which will supply the hardware simulator.

The refresh period T_{ref} = 6666 s during which we must refresh all 4 profiles, which is 1.728 kbit or 216 Byte, makes a rate of:

$$\frac{216}{6666 \times 10^{-6}} = 32.4\ kBps \tag{33}$$

As a SISO channel corresponds to a profile (18taps × 16) bits, or 36 bytes, the rate of 30 MBps allows us to load 7999 SISO channels during the refresh time of 6666 s.

The loading procedure is the same as described in the previous Section for the frequency approach. However, for a FIR filter, the $x(i) \times h(i)$ operation are made for all the impulse response profile at once. Therefore, for an impulse response that has 18 taps and 18 excess delays for one SISO channel, we need to load 36 RAM blocks (Fig. 18). To respect the refresh period, the second profile is saved in the same way on the 36 RAM blocks. Thus, each RAM block contains two profiles.

The signal "Selector" written on 5 bits, controls the demultiplexer which selects one of the 36 RAM block. The signal "Profile In" takes the values "1" and "0" to show which profile is active and used by the FIR filter. The address "Addr w" is the "Profile out". It takes the values "1" and "0" to select the other profile that the new coefficients will be written on.

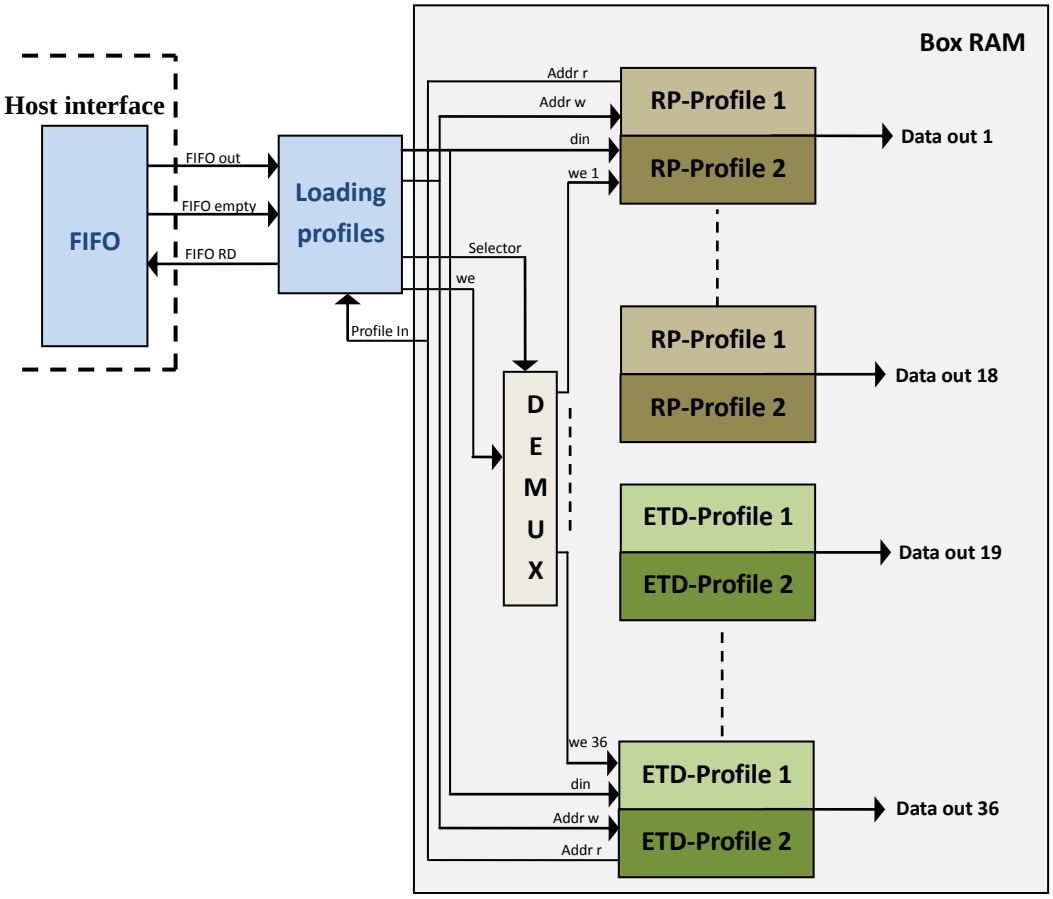


Figure 18. *Loading profiles of impulse responses for SISO channel.*

The reading and writing of the RAM blocks are independent, thus, it is possible to write the new FIR filter coefficients while still reading the old ones. The 36 RAM blocks are loaded and each output "Data out" is directed to a continuous real multiplier where the coefficients are multiplied with the input signal samples contained in the shift register of the FIR filter. The "Addr r" is actually a periodic signal of period twice the sampling period. Thus, all the profiles are charged with the refresh period.

4.2. Implementation on FPGA

The occupancy of the time domain architecture is known after performing operations of synthesis, mapping, place and route from the program written in VHDL. Table 5 shows the device utilization in one Virtex-IV SX35 for one SISO channel using the time domain architecture for the TGn channel model E.

Table 5. *Virtex-IV utilization for SISO time domain architecture for TGn model E*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	986	30,720	4%
Number of LUT	1,355	30,720	5%
Number of occupied slices	593	15,360	4%
Number of DSP block	18	192	10%
Number of RAM block	1	192	1%

Table 6 shows the device utilization in one Virtex-IV SX35 for one SISO channel using the time doamin architecture for the 3GPP-LTE model EVA.

Table 6. *Virtex-IV utilization for SISO time domain architecture for 3GPP-LTE model EVA*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	814	30,720	3%
Number of LUT	1,004	30,720	4%
Number of occupied slices	459	15,360	3%
Number of DSP block	9	192	5%
Number of RAM block	1	192	1%

5. Time Domain versus Frequency Domain Architectures

5.1. Accuracy description

In order to determine the accuracy of the digital block, a comparison is made between the theoretical and the Xilinx output signals. The theoretic output vector of the SISO channel is calculated by:

$$Y_{theory}(t) = \sum_{k=1}^{n_{tap}} h(i_k).x(t - i_k T_s) \tag{34}$$

where is the number of taps of the impulse response and is the vector the taps position.

As we will see in the next section, the Xilinx output and the theoretical output are very close and we can't differentiate them. Thus, we calculated the relative error which is given for each output sample by:

$$RE(i) = \frac{Y_{xilinx}(i) - Y_{theory}(i)}{Y_{theory}(i)}.100 \quad [\%] \tag{35}$$

where Y_{Xilinx} is the vector containing the samples of the Xilinx output signal. The relative SNR is computed by:

$$RSNR(i) = 20\log_{10}\left|\frac{Y_{theory}(i)}{Y_{xilinx}(i) - Y_{theory}(i)}\right| \quad [dB] \tag{36}$$

The global values of the relative error and of the SNR computed for the output signals after the final truncations are necessary to evaluate the accuracy of the architecture. The global relative error is computed by:

$$RE_G = \frac{\|E\|}{\|Y_{theory}\|}.100 \quad [\%] \tag{37}$$

The global SNR is computed by:

$$RSNR_G(i) = 20\log_{10}\left\|\frac{Y_{theory}}{E}\right\| \quad [dB] \tag{38}$$

where $E = Y_{Xilinx} - Y_{theory}$ is the error vector, and for a given vector $X = [\, x_1 , x_2, \ldots, x_L \,]$, its Euclidean norm $\|x\|$ is:

$$\|x\| = \sqrt{\frac{1}{L}.\sum_{k=1}^{L} x_k^2} \tag{39}$$

Fig. 19, 20, 21 and 22 present the output relative error and SNR using the improved frequency architecture and the time domain architecture for TGn channel model E and 3GPP-LTE channel model ETU. The latencies of the architectures are measured from the time where the input signal enters in the ADC and exists from the DAC.

For the improved frequency architecture, the FFT 256 needs 256 cycles to generate its first output sample. Then, the IFFT 256 needs another 256 cycles. Another cycle is needed to the digital adder. Thus, 513 cycles of 165 MHz are needed using WLAN 802.11ac signals and 513 cycles of 50 MHz are needed using 3GPP-LTE channel model EVA. These values are also obtained by ModelSim [23]. It is necessary to add 38 ns of the ADC latency, and 17 ns of the DAC latency, according to their datasheets. In summary, the improved frequency architecture and the converters have a latency, using one SISO channel of TGn channel model E and with WLAN 802.11ac signals, of:

$$\frac{513}{165MHz} + 38ns + 17ns = 3.16\ \mu s \tag{40}$$

and using one SISO channel of 3GPP-LTE channel model EVA and with LTE signals, of:

$$\frac{513}{50MHz} + 38ns + 17ns = 10.31\ \mu s \tag{41}$$

If we consider the absolute delays of the impulse responses and $c = 3.10^8$ m/s is the speed of light, these latencies impose a minimum distance between the transmitter and the receiver of 948 m and 3093 m respectively.

The latency of the time domain architecture is related to the number of taps of the impulse response. The TGn channel model E has 18 taps. Therefore, the FIR filter has 1 cycle of 18 multiplications and 5 addition cycles. Thus, 6 cycles of 165 MHz are needed using WLAN 802.11ac signals. For 3GPP-LTE channel model EVA, which has 9 taps, 5 cycles of 50 MHz are needed using LTE signals. In summary, the time domain architecture and the converters have a latency, using one SISO channel of TGn channel model E and with WLAN 802.11ac signals, of:

$$\frac{6}{165MHz} + 38ns + 17ns = 91.36\ ns \tag{42}$$

and using one SISO channel of 3GPP-LTE channel model

EVA and with LTE signals, of:

$$\frac{5}{50MHz} + 38ns + 17ns = 155\,ns \qquad (43)$$

These latencies impose a minimum distance between the transmitter and the receiver of 27.4 m and 46.5 m.

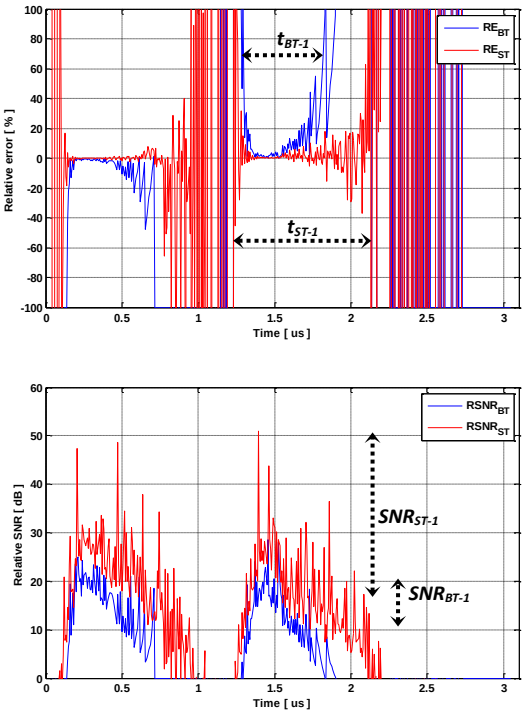


Figure 19. *Improved frequency using TGn model E.*

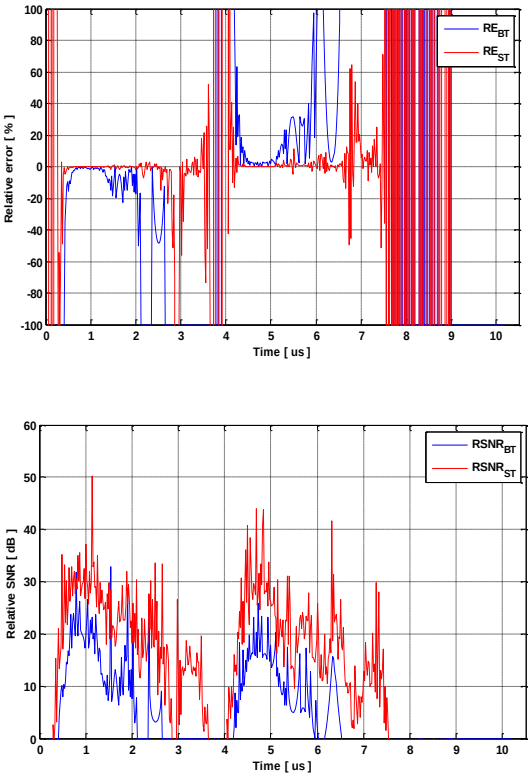


Figure 20. *Improved frequency using 3GPP-LTE model EVA.*

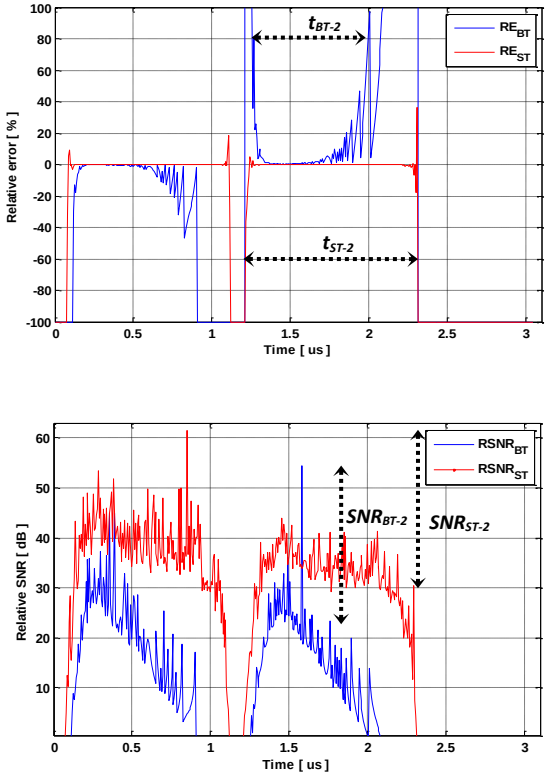


Figure 21. *Time domain using TGn model E.*

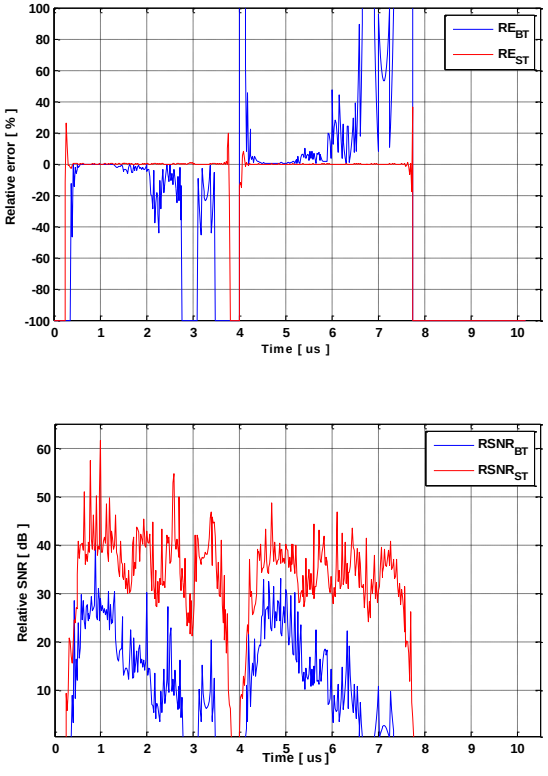


Figure 22. *Time domain using 3GPP-LTE model EVA.*

Fig. 23 presents the latencies obtained by hardware implementation of the time domain architecture using WLAN 802.11ac with TGn channel model E, and Fig. 24 using LTE

with 3GPP-LTE channel model EVA. To measure the latency, we have increased the first tap power and decreased the others to calculate the time of arrival of the output signal.

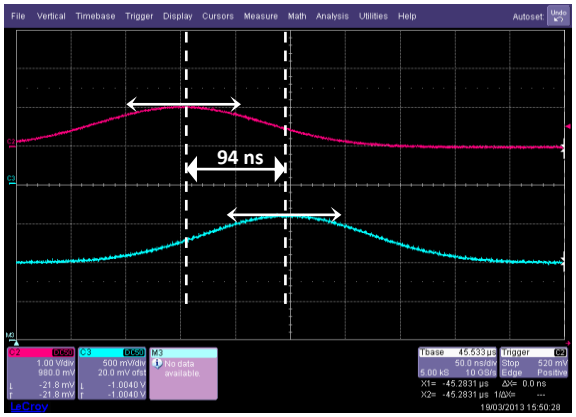


Figure 23. *Latency obtained after hardware implementation for WLAN 802.11ac using TGn model E.*

We notice that the latency obtained is 94 ns for TGn model E with WLAN 802.11ac signals and 162 ns for 3GPP-LTE molel EVA with LTE signals. Thus, with an error of 3 % approximately (due to the delays by the cables), if we compared it to the theory results obtained previously.

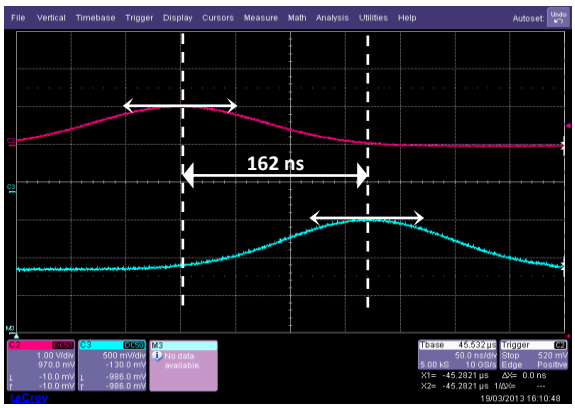


Figure 24. *Latency obtained after hardware implementation for LTE using 3GPP-LTE model EVA.*

5.2. Table of comparison

To present better the results, the global values of the relative error and SNR has to be calculated. Table 7 presents the results and characteristics of the time domain architecture versus the improved frequency architecture.

Three points resume the comparison: the precision of the output signals, the occupation on the FPGA and the latency.

Table 7. *Time domain architecture versus improved frequency architecture.*

	Simulation results	
	Time domain architectrue	***Improved frequency architectrue***
FIR or FFT/IFFTsize	121 (TGn model E); 126 (3GPP-LTE EVA)	256 (TGn model E); 256 (3GPP-LTE EVA)
Number of bits	*x* 14; *h* 16; *y* 14	*x* 14; FFTs IN 16 OUT 16; H 32; IFFTs IN 24 OUT
Occupation (TGn model E)	Slice **4%**; DSP **10%**; RAM **1%**	Slice **26%**; DSP **21%**; RAM **10%**
Occupation (3GPP-LTE model EVA)	Slice **3%**; DSP **5%**; RAM **1%**	Slice **27%**; DSP **21%**; RAM **10%**
Global error (TGn model E)	Using $x_{WLAN}(t)$: **0.01 %** (ST); **0.56 %** (BT)	Using $x_{WLAN}(t)$: **0.46 %** (ST); **2.47 %** (BT)
Global SNR (TGn model E)	Using $x_{WLAN}(t)$: **76 dB** (ST); **44 dB** (BT)	Using $x_{WLAN}(t)$: **46 dB** (ST); **32 dB** (BT)
Global error (3GPP-LTE model EVA)	Using $x_{LTE}(t)$: **0.01 %** (ST); **0.74 %** (BT)	Using $x_{LTE}(t)$: **0.3 %** (ST); **3.78 %** (BT)
Global SNR (3GPP-LTE model EVA)	Using $x_{LTE}(t)$: **75 dB** (ST); **42 dB** (BT)	Using $x_{LTE}(t)$: **50 dB** (ST); **28 dB** (BT)
Latency	**155 *ns* – 91 *ns***	**10.31 *s* – 3.16 *s***

5.2.1. Precision

We start with the precision of the architectures. In the previous figures, we mark $t_{BT\text{-}1}$, $t_{ST\text{-}1}$, $t_{BT\text{-}2}$ and $t_{ST\text{-}2}$ as the margin of the small values of the relative error using the BT and ST for the improved frequency and the time domain architectures respectively. The margin using the BT with the improved frequency architecture is $t_{BT\text{-}1}$=0.53 s, while with the time domain architecture is $t_{BT\text{-}2}$ = 0.75 s. Moreover, using the ST with the improved frequency architecture is $t_{ST\text{-}1}$=0.9 s, while with the time domain architecture is $t_{ST\text{-}2}$=1.2 s. Also we can notice that, in these margins, the relative error is smaller using the time domain architecture, while it present high variations using the frequency domain architecture. The same discussion is made for the relative SNR which is higher using the time domain architecture.

To discuss better the results on all the window of the output signal, the global values of the output signals and SNR are computed and presented in the Table 7. If we compare the global relative errors, using the results of TGn channel model E for example, we notice that the global relative error decreases from 3.78 % using the improved frequency architecture with BT, to 0.74 % using the time domain architecture. Also, using ST, it decreases from 0.3 % using the improved frequency architecture, to 0.01 % using the time domain architecture.

Therefore, after this study, we conclude that the time

domain architecture is more accurate than the improved frequency architecture. Moreover, we notice also that using the ST decreases the error using the time domain architecture from 0.3 % to 0.01%.

Moreover, form a theoretical point of view, the improved frequency architecture use many quantified signals for the input signal, the phase factor of the FFT modules, the output FFT signal, the output of the complex multiplier, the frequency responses for its real and imaginary parts, for the IFFT modules and for the output signals. However, the time domain architecture quantifies only 3 signals: the input signal, the impulse response and the output signal. This is the cause why the time domain architecture has a higher precision.

5.2.1. Occupation on FPGA

According to Table 7, the improved frequency domain architecture presents a slice occupation between 26 and 27 %. However, the time domain architecture occupies 3 to 4 % of slices.

Thus, the improved frequency architecture presents a high slice occupation on the FPGA if we compare it to the time domain architecture. It requires more performing FPGAs to implement high order MIMO channels.

However, in order to simulate an impulse response with more than 192 taps, the new frequency architecture can be used. With a FPGA Vitrex-IV, the size N_F of the FFT/IFFT modules can be chosen up to 65536 in contrast with a FIR filter which is limited to 192 multipliers or DSP blocks on the FPGA, which is a limitation of 192 taps for the impulse response.

5.2.1. Latency

The latency using the time domain architecture is between 91 and 155 *ns*. However, using the improved frequency architecture, it is between 3.16 and 10.31 s.

The latencies using the time domain architecture are way better that the latencies obtained by the improved frequency architecture. In fact, with a FIR filter, the samples are computed together in one stroke, however, with the frequency architectures the samples are obtained after charging the entire coefficient in the FFT/IFFT modules.

6. Adopted Time Domain Architecture

After comparing the previous architectures, we have chosen the time domain architecture which has a better occupation on the FPGA, better latency and better precision.

The comparison of the previous architectures was made using a SISO channel and long input signal to show their validation in the worst conditions. However, after choosing the best architecture, we have to consider more realistic conditions. First, the input signal has to respect the bandwidth chosen between [, +*B*]. Secondly, for the new standards, we have to consider working with MIMO systems. To simplify the tests, a 2×2 MIMO architecture is considered.

6.1. Real input signal

In order to determine the accuracy of the digital block, a comparison is made between the theoretical/Xilinx output signals. An input Gaussian signal $x(t)$ is considered for the two inputs of the 2×2 MIMO simulator. To simplify the calculation, we consider $x_1(t) = x_2(t)$:

$$x(t) = x_1(t) = x_2(t) = x_m e^{-\frac{(t-m_x)^2}{2\sigma_x^2}}, \quad 0 \le t \le W_t \tag{44}$$

In fact, the FT of a Gaussian signal is also Gaussian signal, and to obtain a signal $x(t)$ that respect the bandwidth [, +*B*], the following steps are considered:

In frequency domain, the Gaussian input signal = *FT* is computed by:

$$X(f) = x_m \sigma_x \sqrt{2\pi} e^{-2\pi^2 f^2 \sigma_x^2} e^{-j2\pi f m_x} \tag{45}$$

with

$$|X(f)| = x_m \sigma_x \sqrt{2\pi} e^{-2\pi^2 f^2 \sigma_x^2} \tag{46}$$

This signal spectrum is limited between and +*B* if:

$$6\sigma_X \le B \tag{47}$$

where is the standard deviation of .

Comparing the first and the third equation, we obtain:

$$2\pi\sigma_X\sigma_x = 1 \tag{48}$$

Thus, that corresponds to the considered band of the standard used, is obtained:

$$\sigma_x \ge \frac{3}{\pi B} \tag{49}$$

To obtain *x(t)* centered between [, +*B*], it must be multiplied by:

$$x(t) \to x(t).\cos\left(2\pi.\left(\frac{B}{2}+\Delta\right).t\right) \tag{50}$$

In our work, we considered = $3/\pi B$. m_x is chosen equal to $20T_s > 3$ for both WLAN 802.11ac and LTE signals. Moreover, << *B* is chosen equal 2 MHz. These values are small enough to show the effect of each tap on the output signal. For WLAN 802.11ac, B = 80 MHz and $T_s = 1/f_s$ = 6 *ns*. Thus, we obtain = $2T_s$. This signal is named $x_{WLAN}(t)$ and is presented in Fig. 25.

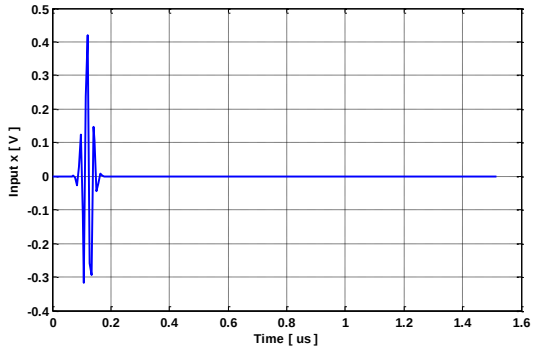


Figure 25. *Input signal for WLAN 802.11ac.*

For LTE, B = 20 MHz and $T_s = 1/f_s$ = 20 ns. Thus, we obtain = $2.5T_s$. This signal is named $x_{LTE}(t)$ and is presented in Fig. 26.

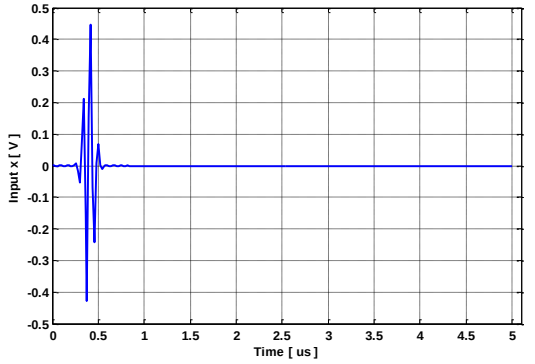


Figure 26. *Input signal for LTE.*

6.2. 2×2 MIMO Architecture

Four FIR filters are considered to simulate 2×2 MIMO channels. For each SISO channel, the FIR length and the number of used multipliers are determined by the non-null taps of the impulse responses. To use a limited number of multipliers on the FPGA, the delays addresses are controlled by connecting each multiplier block of the FIR by the corresponding shift register block. Thus, the number of multipliers in the FIR filters is equal to the maximum number of non-null taps.

The theoretic output signals of a 2×2 MIMO channel are calculated by:

$$y_1(t) = \sum_{k=1}^{N_t} h_{11}(i_k).x_1(t - i_k T_s) + \sum_{k=1}^{N_t} h_{21}(i_k).x_2(t - i_k T_s) \quad (51)$$

$$y_2(t) = \sum_{k=1}^{N_t} h_{12}(i_k).x_1(t - i_k T_s) + \sum_{k=1}^{N_t} h_{22}(i_k).x_2(t - i_k T_s) \quad (52)$$

N_t is the number of taps of the impulse response. $h_q(i_k)$ is the attenuation of the k^{th} path with the delay $i_k T_s$.

Figure 27 presents 2×2 MIMO time domain architecture based on 4 FIR filters with N_t = 18 multipliers. We have developed our own FIR filter instead of using Xilinx MAC FIR filter to make it possible to reload the FIR filter coefficients. The number of bits at the output before the truncation is computed by:

$$M_y = M_h + M_x + M_{MIMO} + M_T \quad (53)$$

where M_{MIMO} is computed by:

$$M_{MIMO} = \lceil \log_2 N_{TX} \rceil \quad (54)$$

where in our case M_{MIMO} = 1 bit for 2×2 MIMO system (for the sum: $y_{11}+y_{21}$ and $y_{12}+y_{22}$).

6.3. Occupation on FPGA

As the development board has 2 ADC and 2 DAC, it can be connected to only 2 down-conversion and 2 up-conversion RF units. Four FIR filters are needed to simulate a one-way 2×2 MIMO radio channel. The occupancy of the time domain architecture is known after performing operations of synthesis, mapping, place and route from the program written in VHDL. Table 8 shows the device utilization in one Virtex-IV SX35 for 2×2 MIMO channel using the time domain architecture for the TGn channel model E.

Table 8. *Virtex-IV utilization for 2×2 MIMO time domain architecture for TGn model E*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	3,992	30,720	13%
Number of LUT	5,526	30,720	18%
Number of occupied slices	2,440	15,360	16%
Number of DSP block	72	192	38%
Number of RAM block	1	192	1%

Table 9 shows the device utilization in one Virtex-IV SX35 for 2×2 MIMO channel using the time domain architecture for the 3GPP-LTE model EVA.

Table 9. *Virtex-IV utilization for 2×2 MIMO time domain architecture for 3GPP-LTE model EVA*

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of flip flops	3,296	30,720	11%
Number of LUT	4,097	30,720	14%
Number of occupied slices	1,891	15,360	13%
Number of DSP block	36	192	19%
Number of RAM block	1	192	1%

We notice that the occupation of slice on the FPGA of a 2×2 MIMO system is 16 % for the TGn channel model E and 16 % for the 3GPP-LTE model EVA. In fact, these occupations are equal to the occupations of a SISO channel multiplied by four and with additional slices added because of the two digital adders that operates $y_{11} + y_{21}$ and $y_{12} + y_{22}$. Moreover, the 2×2 MIMO system has small occupation on the FPGA Virtex-IV. In fact, we can implement up to 4×4 MIMO system in the FPGA for the 3GPP-LTE model EVA (because for TGn channel model E the number of multiplier is equal to 18×(4×4) = 288>192). However, we are limited by the 2 ADC and the 2 DAC.

6.3. Results and accuracy

Table 10 shows the global values of the relative error and SNR for the considered 2×2 MIMO time domain architecture of the TGn channel model E and 3GPP-LTE channel model EVA.

Table 10. *Global relative error and SNR for 2×2 MIMO time domain architecture.*

	Error (%)		***SNR (dB)***	
	y_1	y_2	y_1	y_2
	TGn model E with $x_{WLAN}(t)$			
ST	0.0334	0.0328	69.52	69.68
BT	3.9758	3.9435	28.01	28.09
	3GPP-LTE model EVA with $x_{LTE}(t)$			
ST	0.0362	0.0382	68.82	68.35
BT	2.9263	4.1348	30.67	27.68

Fig. 28, 29, 30 and 31 presents the Xilinx output signal, the relative error and the relative SNR for $y_1(t)$ and $y_2(t)$ using TGn model E with $x_{WLAN}(t)$ at the input and 3GPP-LTE channel model EVA using $x_{LTE}(t)$, for the 2×2 MIMO time domain architecture.

We can see that the benefit of a ST in the case of using a real signal that respects the band $[\Delta, \Delta + B]$. Also, as we can see from the figure of the relative errors that the ST provide a low variation around zero in the margin of time where the output signal is high. However, the BT presents high variations of the relative error and on lower time margin.

The latency of the 2×2 MIMO time domain architecture is calculated in the same as previously, however, one additional cycle is needed to sum the outputs for the 2×2 MIMO system. Thus, in summary, the time domain architecture and the converters have a latency, using 2×2 MIMO channel of TGn channel model E and with WLAN 802.11ac signals, of:

$$\frac{7}{165MHz} + 38ns + 17ns = 97.42 \ \ ns \tag{55}$$

And using 2×2 MIMO channel of 3GPP-LTE channel model EVA and with LTE signals, of:

$$\frac{6}{50MHz} + 38ns + 17ns = 175 \ \ ns \tag{56}$$

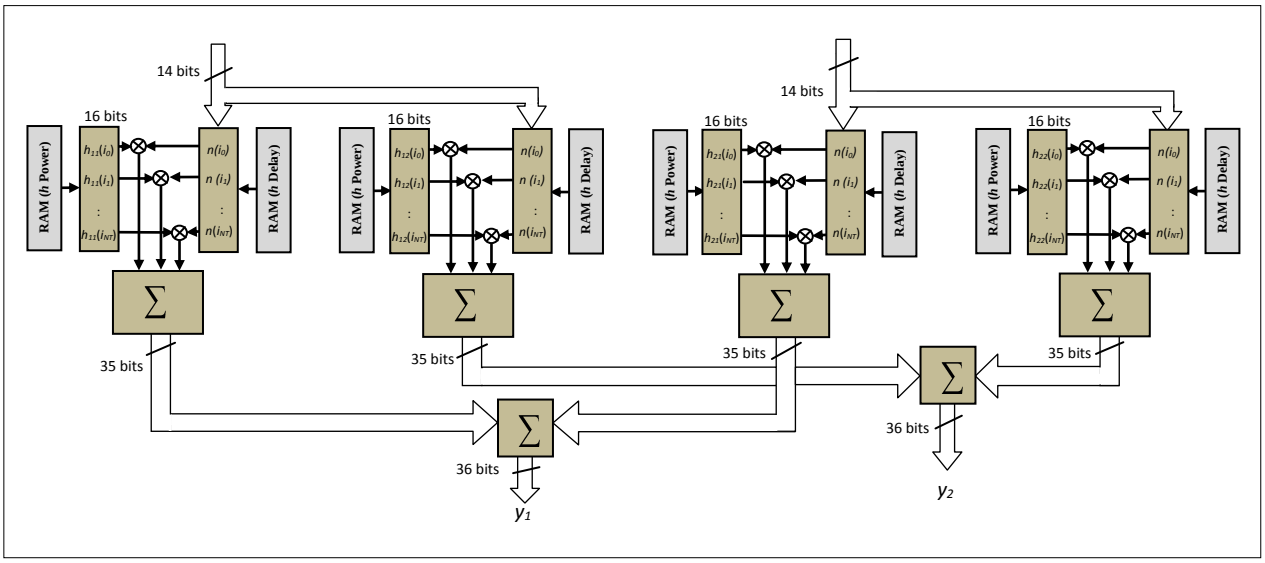


Figure 27. *2×2 MIMO time domain architecture.*

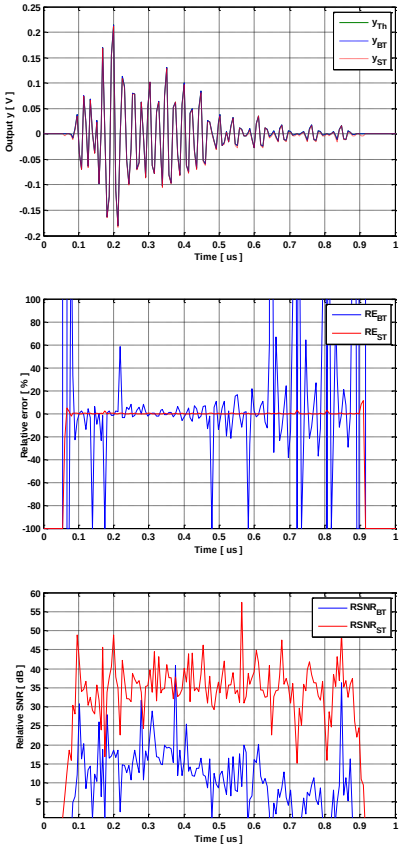


Figure 28. *Results for y_1 using TGn model E.*

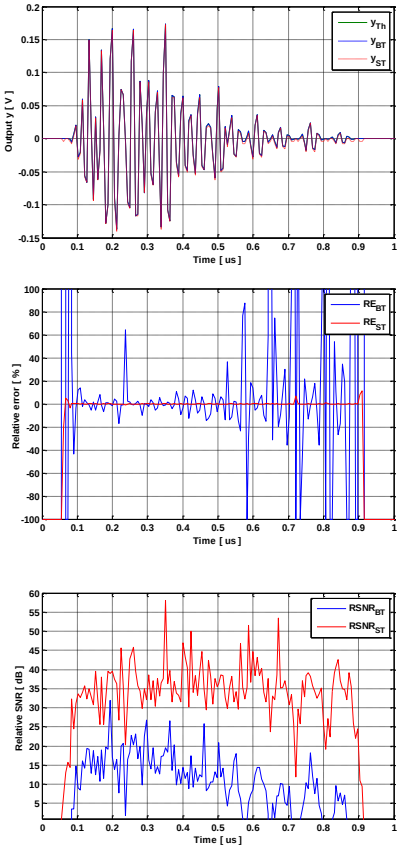


Figure 30. *Results for y_2 using TGn model E.*

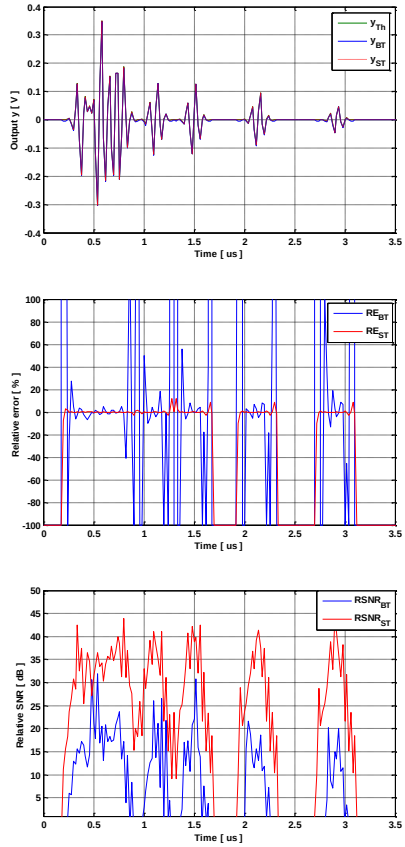


Figure 29. *Results for y_1 using 3GPP-LTE model EVA.*

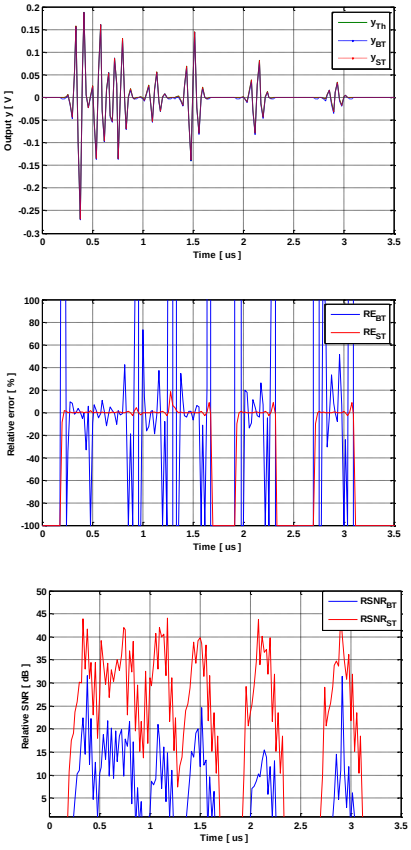


Figure 31. *Results for y_2 using 3GPP-LTE model EVA.*

7. Conclusion

In this paper, the frequency approach has been presented and analyzed in detail. First, the simple frequency architecture has been studied. Each block that compose it, from the FFT/IFFT modules to the multiplier, the truncation, the memory and the convertors, has been presented, analyzed and detailed. The size of the FFT/IFFT modules depends on the last excess delay of the impulse response. After that, the entire simple SISO frequency architecture has been implemented on the FPGA. It has been tested with Gaussian input signal that have limited duration in time and frequency domains. Its occupation on the FPGA (12 % of used slices), latency and accuracy have been analyzed. It has been shown that the ST reduces the relative error of the output signal significantly. After testing the simple frequency architecture with long input signal that has a duration larger than the size of the FFT/IFFT blocks, it has been shown that this architecture gives wrong output results. Therefore, we analyzed an improved frequency architecture that works for input signals in streaming mode. The improved frequency architecture for a SISO channel has been implemented on the FPGA and the results were provided. It has been shown that the improved frequency architecture has very high occupation (27 % of used slices) on the FPGA. Therefore, it is impossible to implement a 2×2 MIMO system using this architecture on an FPGA Virtex-IV.

The time domain architecture of the digital part of the hardware simulator has also been analyzed. The blocks that compose it form the FIR filter (which contains the multiplier, the shift register and the memory) to the truncation, have been presented, analyzed and detailed. The number of multipliers used depends on the number of non-null tap of the impulse response. After that, the entire simple SISO time domain architecture has been implemented on the FPGA and the results were provided. A comparison between the time domain architecture and the improved frequency architecture has been made with the same input signal and for the same channel. It has been show that the time domain architecture has a better occupation on the FPGA (4 % of occupied slices instead of 27 % using the improved frequency architecture), a better latency (of 155 ns instead of 10.31 µs using the improved frequency architecture), and better precision (up to 76 dB instead of 46 dB using the improved frequency architecture). The comparison of the previous architectures was made using a SISO channel and long input signal to show their validation in the worst conditions. However, after choosing the best architecture which is the time domain architecture, we have considered more realistic conditions. First of all, the input signal has to respect the bandwidth chosen between [, +*B*]. Secondly, for the new standards, we have considered working with 2×2 MIMO systems. The channel has been simulated using these two conditions.

For our future work, simulations made using a Virtex-VII [3] XC7V2000T platform will allow us to simulate up to 300 SISO channels. In parallel, measurement campaigns will be carried out with the MIMO channel sounder realized by IETR to obtain the impulse responses of the channel for specific and various types of environments. The final objective of these measurements is to obtain realistic MIMO channel models in order to supply the hardware simulator. A graphical user interface will also be designed to allow the user to reconfigure the simulator parameters.

Acknowledgements

This work is a part of CEDRE program and PALMYRE-II project with the support of “Region Bretagne”.

References

[1] A. S. Behbahani, R. Merched, and A. Eltawil, “Optimizations of a MIMO relay network,” IEEE, vol. 56, no. 10, pp. 5062–5073, Oct. 2008.

[2] B. A. Cetiner, E. Sengul, E. Akay, and E. Ayanoglu, “A MIMO system with multifunctional reconfigurable antennas,” IEEE Antennas Wireless Propag. Lett., vol. 5, no. 1, pp. 463–466, Dec. 2006.

[3] "Xilinx: FPGA, CPLD and EPP solutions", www.xilinx.com.

[4] Wireless Channel Emulator, Spirent Communications, 2006.

[5] Baseband Fading Simulator ABFS, Reduced costs through baseband simulation, Rohde & Schwarz, 1999.

[6] P. Murphy, F. Lou, A. Sabharwal, P. Frantz, "An FPGA Based Rapid Prototyping Platform for MIMO Systems", Asilomar Conf. on Signals, Systems and Computers, ACSSC, vol. 1, pp. 900-904, 9-12 Nov. 2003.

[7] P. Murphy, F. Lou, J. P. Frantz, "A hardware testbed for the implementation and evaluation of MIMO algorithms", Conf. on Mobile and Wireless Communications Networks, Singapore, 27-29 Oct. 2003.

[8] V. Erceg et al., “TGn Channel Models”, IEEE 802.11-03/940r4, May 10, 2004.

[9] Agilent Technologies, “Advanced design system – LTE channel model - R4-070872 3GPP TR 36.803 v0.3.0”, 2008.

[10] G. El Zein, R. Cosquer, J. Guillet, H. Farhat, F. Sagnard, “Characterization and modeling of the MIMO propagation channel: an overview” European Conference on Wireless Technology, ECWT, 2005.

[11] H. Farhat, R. Cosquer, G. El Zein, "On MIMO channel characterization for future wireless communication systems", 4G Mobile & Wireless Comm.Tech., River Publishers, Aalborg, Denmark, 2008.

[12] B. Habib, H. Farhat, G. Zaharia, G. El Zein, “Hardware Simulator Design for MIMO Propagation Channel on Shipboard at 2.2 GHz”, Springer, Journal of Wireless Personal Communications, 2012, doi: 10.1007/s11277-012-0954-2.

[13] S. Picol, G. Zaharia, D. Houzet and G. El Zein, "Hardware

simulator for MIMO radio channels: Design and features of the digital block", Proc. of IEEE VTC-Fall 2008, Calgary, Canada, 2008.

[14] S. Picol, G. Zaharia, D. Houzet and G. El Zein, "Design of the digital block of a hardware simulator for MIMO radio channels", IEEE PIMRC, Helsinki, Finland, 2006.

[15] D. Umansky, M. Patzold, "Design of Measurement-Based Stochastic Wideband MIMO Channel Simulators", IEEE Globecom, Honolulu, Hawai, 30 Nov. - 4 Dec. 2009.

[16] H. Eslami, S.V. Tran and A.M. Eltawil, "Design and implementation of a scalable channel Emulator for wideband MIMO systems", IEEE Trans. on Vehicular Technology, vol. 58, no. 9, pp. 4698-4708, Nov. 2009.

[17] S. Fouladi Fard, A. Alimohammad, B. Cockburn, C. Schlegel, "A single FPGA filter-based multipath fading emulator", VTC-Fall, Canada, 2009.

[18] B. Habib, G. Zaharia and G. El Zein, "MIMO Hardware Simulator: New Digital Block Design in Frequency Domain for Streaming Signals", Journal of Wireless Networking and Communications, Vol. 2, No. 4, 2012, pp. 55-65.

[19] W. C. Jakes, “Microwave mobile communications”, Wiley & Sons, New York, Feb. 1975.

[20] J. P. Kermoal, L. Schumacher, K. I. Pedersen, P. E. Mogensen, F. Frederiksen, “A stochastic MIMO radio channel model with experimental validation”, IEEE Journal on Selected Areas of Commun., Vol. 20, No. 6, Aug. 2002, pp. 1211-1226.

[21] Q. H. Spencer, et al., “Modeling the statistical time and angle of arrival characteristics of an indoor environment”, IEEE J. Select. Areas Commun., Vol. 18, No. 3, March 2000, pp. 347-360.

[22] C-C. Chong, D. I. Laurenson, S. McLaughlin, “Statistical Characterization of the 5.2 GHz wideband directional indoor propagation channels with clustering and correlation properties”, in proc. IEEE Veh. Technol. Conf., Vol. 1, Sept. 2002, pp. 629-633.

[23] “ModelSim - Advanced Simulation and Debugging”, http://model.com.