Journal of Engineering and Sustainable Development
www.jeasd.org
Vol. 21, No.01, January 2017
ISSN 2520-0917

# FPGA BASED 2×2 MMSE MIMO-OFDM SYSTEM USING XILINX SYSTEM GENERATOR

Dr. Fadhil Sahib Hasan *

Lecturer, Electrical Engineering Department, Al-Mustansiriayah University, Baghdad, Iraq

**Abstract:** Multiple-input multiple-output orthogonal frequency division multiplexing (MIMO-OFDM) is a powerful technique to increase the capacity of wireless communication system and decrease the effect of selective fading to flat fading channel. In this paper 2×2 MIMO-OFDM system is implemented using Xilinx system generator (XSG). Simple MMSE equalizer is implemented at the receiver to detect the signal over MIMO channel. The following features are implemented and added to the proposed system: increasing security of the system using chaos based scrambling, using 16-ADQAM modulation to solve the ambiguity problem and implement FFT in pipelining method. The results show that the original data is recovered successfully at the receiver. The VHDL code file is generated for this system for Xilinx Virtex-4 device for hardware implementation purposes. The system is routed in successfully using ISE 14.1 program with resources of 21% slices Flip Flop, 57% LUT, 71% occupied slices and 87% DSP 48s numbers from the selected device. The Xilinx system generator is more flexible, easier, and reliable and gives optimum design for FPGA technique comparing with conventional FPGA design.

**Keywords:** *MIMO, OFDM, MMSE, Chaos based PRBG, ADQAM, Xilinx System Generator, FPGA.*

## أعتماد FPGA في بناء منظومة 2× 2 MMSE MIMO-OFDM باستخدام مولد النظام Xilinx

**الخلاصة:** تعتبر تقنية المقسم المتعامد التردد ذو المدخلات المتعددة والمخرجات المتعددة اسلوب فعال لزيادة قدرة الاستيعاب في انظمة الاتصال اللاسلكية وتقليل تأثير التوهين الانتقائي للقناة الى توهين مسطح. في هذا البحث تم بناء منظومة $2 \times 2$ MIMO $-$ OFDM بأستخدام مولدة نظام Xilinx . تم استخدام تقنية MMSE Equalizer لكشف الاشارة خلاة قناة MIMO. تم اضافة الخصائص التالية للمنظومة: زيادة امنية النظام باستخدام التشفير القائم على الفوضوية، حل مشكلة الغموض باستخدام التضمين 16ADQAM و بناء الدالة FFT باستخدام طريقة الانابيب. النتائج اثبتت بان الاشارة الاصلية تم استرجاعها عند المستلم بشكل صحيح ولقد تم توليد ملف الشفرة VHDL للنظام المقترح مع جهاز Xilinx Virtex-4. تم توجيه النظام بنجاح باستخدام برنامج ISE 14.1 بموارد 21 ٪ شرائح Flip Flop ، 57 ٪ LUT ، 71 ٪ شرائح المحتلة و 87 ٪ DSP 84s من الجهاز المحدد. ان مولد النظام Xilinx يعتبر اكثر مرونة ،سهولة ، موثوقية ويعطي التصميم الامثل لتقنية FPGA مقارنه بالطريقة التقليديه لتصميم FPGA.

## 1. Introduction

Orthogonal Frequency Division Multiplexing (OFDM) is a multi-carrier transmission. This technique transform a frequency selective channel into a group of flat narrowband channels that it is immunity against large delay spreads of the wireless channel by preserving orthogonality in the time and frequency domain[1]. This

*fadel_sahib@yahoo.com

technique can be integrated with multiple input multiple output (MIMO) system to increase either communication diversity as space time block coding (STBC) [2]or capacity as spatial multiplexing (SM)[3].

Field programmable gate array (FPGA) is widely used in communication system because it enhances the processing speed, explored the parallelism in implementation and reduces the cost of the design [1]. In the recent years, spatial multiplexing MIMO-OFDM implementation by using FPGA technique becomes the most interest for the researchers [4-11]. Kerttula [4] designed and implemented 2×2 MIMO-OFDM system with List sphere detector (LSD) algorithm using FPGA. Yoshizawa [5] designed minimum mean square error (MMSE) MIMO detection using pipeline processing for 2×2 and 8×8 MIMO-OFDM system. In the same year, Park [6] designed Pipelining processing for fast Fourier transform (FFT) an applied to MIMO-OFDM system. In [7] Chen implemented a 4×4 MIMO-OFDM software defined radio (SDR) system using MMSE detector and convolutional code. MIMO channel estimation in MIMO-OFDM system was designed and implementation in [8] and [9].Babu [10] optimized MIMO-OFDM system using convolutional code and Viterbi decoder. Sathya [11] designed and implemented Fixed Sphere Decoding (FSD) to reduce complexity in MIMO OFDM detection.

In this paper 2×2 MMSE MIMO OFDM is implemented using Xilinx system generator. Three points are added to enhance the system: increase the security using chaos stream ciphering, use Angle differential QAM (ADQAM) to solve the ambiguity problem and implement fast Fourier transform (FFT) in pipelining manner. Also the system can be designed in optimal and flexible way using system generator (SG) tools.

## 2. Xilinx System Generator (XSG)

System generator (SG) is the one of useful Xilinx DSP design tool that provides over 90 blocks of DSP such as multiplexer, shift register, adder, accumulator and multipliers for implementing specific application. Also provides complex DSP blocks such as convolutional encoder, FIR filter and FFT. Designer in SG is thinking in the same manner as Matlab Simulink environment and can be integrated with Simulink block set to test or help in the design in friendly way. When the system is implemented using Xilinx system generator the optimum hardware description language (HDL) netlists or Bitstreams compilation file can be generated directly including synthesis and Xing flow routing to be used for programming the FPGA device using Integrated System Environment (ISE) design suite program. A testbench representation can be generated in Matlab Simulink environment for use with Xilinx ISE simulator or ModelSim. XSG can be specified at the hardware design details for the specific FPGA Xilinx device [18]. Fig. 1 shows the SG design flow.

The advantage of using System Generator can be summarized as: reducing the time spent by the designer for simulation of the system, flexibility due to update the design parameters in quickly and test the effect on the system, The HDL files of the system can be generated using compilation for ISE and Xilinx FPGAs, XSG blockset can be

integrated with Matlab Simulinlk blocks for helping and co-simulating the system designing, the design can be implemented in optimal way with minimum cost [18].

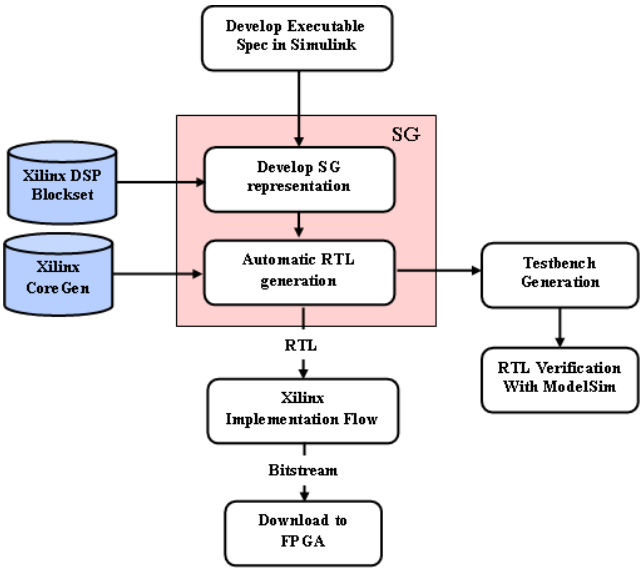



Figure 1. Design flow of SG [18].

## 3. Mmse Mimo-Ofdm System

The proposed 2×2 MMSE MIMO OFDM system is shown in Fig. 2. In the first, the bit streams are scrambled using chaos technique to increase the security of information. Then convolutional code is used to enhance error performance of the system. Block interleaving is take place after that to delete the affect of burst errors, block interleaving is used here for its simplicity. To overcome ambiguity problem in 16-QAM, 16-ADQAM instead is used. Parsing makes data stream to be separated into multiple independent channel each one is passing through 16 points IFFT and cyclic prefix with 4 points. 2×2 MIMO flat fading channel and AWGN are take place to produce the received signal where MMSE MIMO detection is used at receiver side to detect the multiple received data after remove cyclic prefix guard and FFT are taken. All other blocks in the receiver side are designed in inverse mode to that at transmitter side to recover the original stream bits. We assume here the MIMO channel parameters are perfect estimated at input of MIMO detection. The detail description of MIMO OFDM blocks are summarized in next subsections.

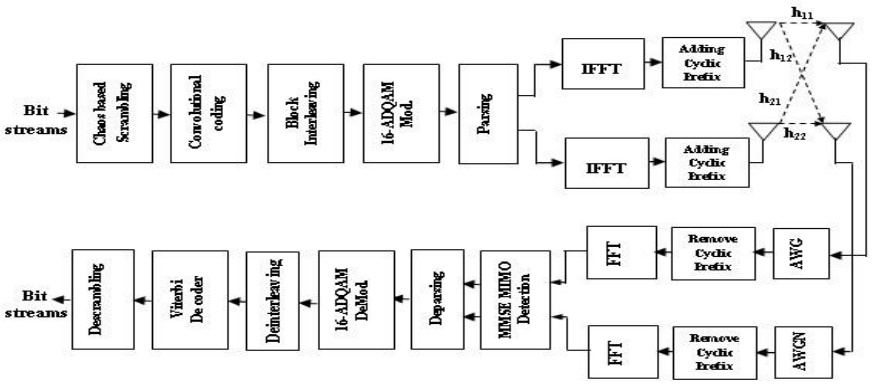



Figure 2. MMSE MIMO-OFDM system

### 3.1. *Chaos Based Scrambling*

The security of communication system can be increased by encrypt or scramble the data source before sending across the channel. One of the popular encryption techniques for digital communication is stream cipher. The stream cipher system depends on pseudo random bit generator (PRBG) XOR with data source to produce the ciphered message at the receiver side the original message can be deciphered by XOR with the same key bit generator that is synchronized with that at transmitter side. PRBG can be generated using linear feedback shift register (LFSR) [12] or using the chaos technique.

Chaos based PRBG is used in the last years for increasing the security of the systems like communication and encryption message due to have more randomness and security than LFSR [13].

In this paper, PRBG can be generated using two logistic maps that were proposed in [14]. The simple mathematical model of the logistic map is expressed as [14]:

$$X_{n+1} = \alpha X_n (1 - X_n) \quad (1)$$

where $X_n$ is a state variable, which lies in [0,1] values and $\alpha$ is the system parameter. In this paper $\alpha = 4$. Figs. 3 and 4 show block diagram for chaos based PRBG and scrambling system respectively. Two Chaotic logistic maps are used to generate PRNG each with different initial conditions, $X_0$=0.939243 and $Y_0$=0.162343 for the first and second generation respectively.The PRBG at receiver side is identical to that at transmitter side. The key stream in the transmitter and receiver sides must be synchronized to recover the original data source.

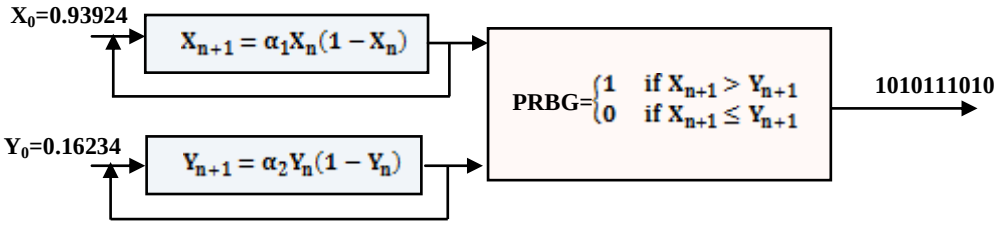



Figure 3. Block diagram of Chaos based PRBG

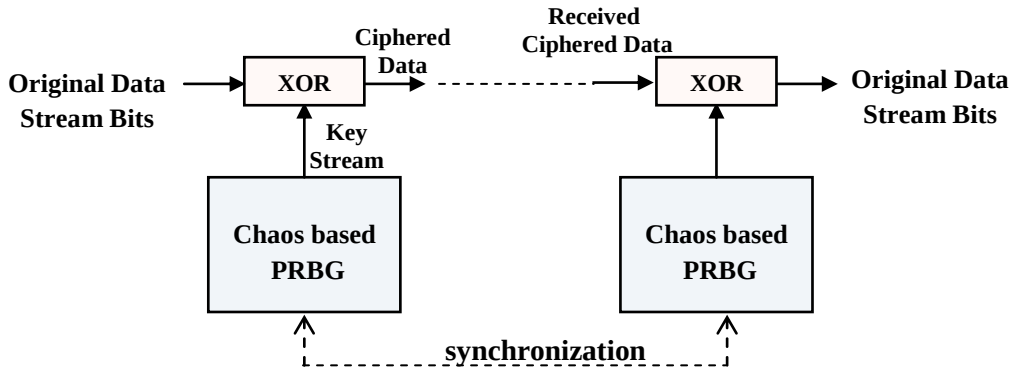



Figure 4. Block diagram of Chaos based scrambling System

### 3.2. *Angle Differential Quadrature Amplitude Modulation (16-ADQAM)*

The widely used in digital modulation is 16-QAM (Quadrature Amplitude Modulation) because it has good power and spectral efficiency. Although the square QAM has disadvantaged that it suffers from ambiguity problem. Therefore angle differential QAM (ADQAM) was studied in [15] is used to solve this problem. This study showed that 16-ADQAM has degraded in SNR about 0.5 dB only when compared with coherent QAM over AWGN channel. In this paper, 16-ADQAM is implemented using Xilinx system generator (XSG).

16-ADQAM requires four bits group and there are $2^4$ possible transitions between two consecutive transmitted symbols s(i) and s(i-1). The transmitted symbol s(i) can be expressed in terms a quadrant center C(i), and a displacement D(i) as [15]:

$$s(i)=C(i)+D(i) \quad (2)$$

$$C(i)=C(i-1)\, e^{j\Delta\theta_1(i)} \quad (3)$$

$$D(i)=D(i-1)\, e^{j\Delta\theta_2(i)} \quad (4)$$

where $\Delta\theta_1$ and $\Delta\theta_2$ are the differential angles which can be determined from the first two bits ($b_0$,$b_1$) and second two bits ($b_2$,$b_3$)of QAM symbol respectively as shown in Table 1.

Table 1. Dibit to differential angle mapping for $\Delta\theta_1$ and $\Delta\theta_2$

| *Dibit* | **Θ** |
|---|---|
| 0 0 | 0 |
| 0 1 | $\pi/2$ |
| 1 1 | $\Pi$ |
| 1 0 | $3\pi/2$ |

s(0) is initialized by assuming $C(0)=R_1e^{j\pi/4}$ and $D(0)=R_2e^{j\pi/4}$, where $R_1=2\sqrt{2}$ represents the distance between the origin point and the quadrant center point, and $R_2=\sqrt{2}$ is the distance between the quadrant center point and the constellation point.

At the receiver, the symbols s(i) are corrupted with AWGN , n(i). The receiver symbols, y(i) can be written as [15]:

$$y(i)=s(i)\, e^{j\emptyset}+ n(i) \quad (5)$$

where $\emptyset$ is the phase ambiguity which takes any value between 0 and $3\pi/2$ in step $\pi/2$. A two stage differential decoding is used for 16-ADQAM demodulation. Substituting Equation (2), get:

$$y(i)=C(i)\ e^{j\emptyset} + D(i)\ e^{j\emptyset} + n(i)$$
$$=C_r(i) + D_r(i) + n(i) \quad (6)$$

where $C_r(i)$ and $D_r(i)$ are the quadrant center and displacement symbols that are rotated by angle Ø respectively. $C_r(i)$ can be detected from the received symbols, y(i) as [15]:

$$\ddot{C}_r(i) = R_1 \times [\text{sgn}(\text{real}[y(i)]) + j\,\text{sgn}(\text{imag}[y(i)])] \quad (7)$$

where sgn (.) is the signum function, real (.) is the real part of a complex signal and imag(.) is the imaginary part. The differential angle $\Delta\theta_1$ can be detected from the present and previous values of $\hat{C}_r(i)$ as [15]:

$$\Delta\theta_1(i) = \begin{cases} 0 & \text{if } \hat{C}_r(i) \times \left(\hat{C}_r(i-1)\right)^* = R_1{}^2 \\ \frac{\pi}{2} & \text{if } \hat{C}_r(i) \times \left(\hat{C}_r(i-1)\right)^* = jR_1{}^2 \\ \pi & \text{if } \hat{C}_r(i) \times \left(\hat{C}_r(i-1)\right)^* = -R_1{}^2 \\ \frac{3\pi}{2} & \text{if } \hat{C}_r(i) \times \left(\hat{C}_r(i-1)\right)^* = -jR_1{}^2 \end{cases} \quad (8)$$

$D_r(i)$ can be detected from the difference between y(i) and $\hat{C}_r(i)$ as [15]:

$$\hat{D}_r(i) = R_2 \times [\text{sgn}(\text{real}[y(i) - \hat{C}_r(i)]) + j\,\text{sgn}(\text{imag}[y(i) - \hat{C}_r(i)])] \quad (9)$$

The differential angle $\Delta\theta_2$ can be detected from the present and previous values of $\hat{D}_r(i)$ as [15]:

$$\Delta\theta_2(i) = \begin{cases} 0 & \text{if } \hat{D}_r(i) \times \left(\hat{D}_r(i-1)\right)^* = R_2{}^2 \\ \frac{\pi}{2} & \text{if } \hat{D}_r(i) \times \left(\hat{D}_r(i-1)\right)^* = jR_2{}^2 \\ \pi & \text{if } \hat{D}_r(i) \times \left(\hat{D}_r(i-1)\right)^* = -R_2{}^2 \\ \frac{3\pi}{2} & \text{if } \hat{D}_r(i) \times \left(\hat{D}_r(i-1)\right)^* = -jR_2{}^2 \end{cases} \quad (10)$$

A two stage decoding of 16-ADAM system is shown in Fig. 5.

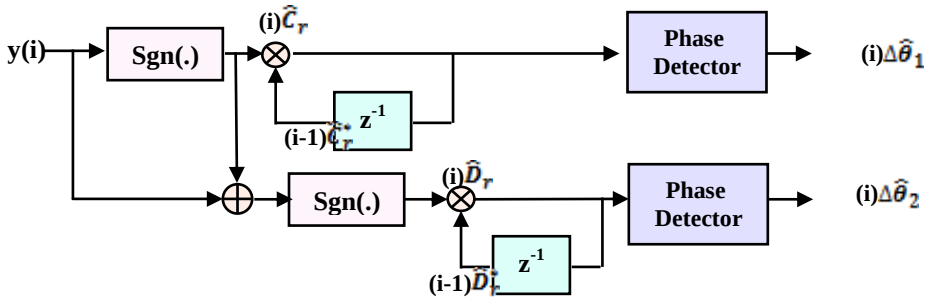



Figure 5. Block diagram of 16-ADQAM detection.

### *3.3 Fast Fourier Transform (FFT)*

16 points decimation in time fast Fourier Transform (FFT) algorithm can be implemented by pipelining techniques in which different functions can be executed at the same time therefore the throughput is increased [16]. However, the pipelining required more hardware than to that of a system without pipelining. Fig. 6 shows flow diagram of 16-point decimation in time FFT algorithm. To implement 16 points FFT, four stages are required {stage_1, stage_2, stage_3 and stage_ 4} in this modules only addition and subtraction operation required. Also three multiple modules {Multiple_1, Multiple_2, and Multiple_3}is required where in this modules the incoming complex samples from the previous module is multiplied by twiddle factor $w_{16}^{n} = e^{j\frac{2\pi n}{16}}$, $n = \{0,1,2,3,4,5,6,7\}$.Inverse fast Fourier transform (IFFT) is implemented in the same manner only twiddle factor is replaced by $w_{16}^{-n} = e^{-j\frac{2\pi n}{16}}$. For Stage_1 module there are 8 functions of 2 points each with two complex additions. For Stage_2 module there are 4 functions of 4 points each with 4 complex additions. For Stage_3 module there are 2 functions of 8 points each with 8 complex additions. Stage_4 module has one function of 16 points with 16 complex additions. Each of Mutilple_1, Mutilple_2 and Mutilple_3 has 8 complex multiplications. The total complex additions is $16\times\log_2(16)=64$ and the total complex multiplications is $(16/2)\times\log_2(16)=32$.

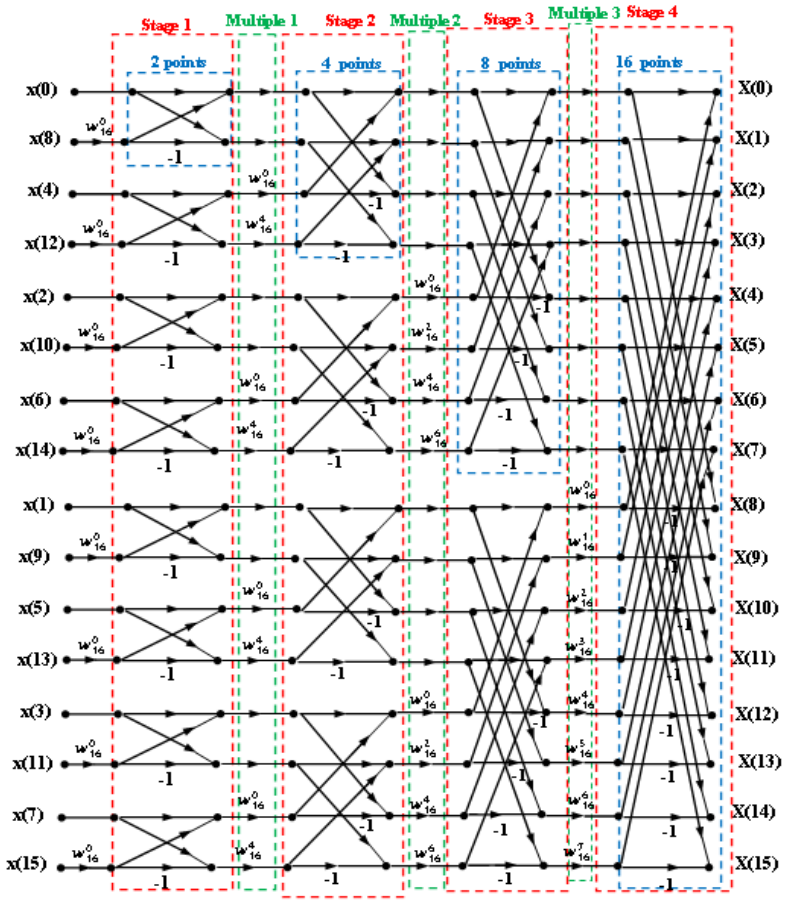



Figure 6. 16-points decimation in time FFT algorithm [19].

### *3.4 Mimo Detection*

Spatial multiplexing (SM) technique increases the capacity gain linearly by increasing number of transmitter and receiver antennas without requiring any additional spectral resources. The relation between the transmitted and received signal for $N_t$ transmitter antennas and $N_r$ receiver antennas MIMO channel is given by [5]:

$$y = Hx + n \tag{11}$$

where $x=[x_1, x_2,\ldots, x_{N_t}]^T$ and $y=[y_1, y_2, \ldots, y_{N_r}]^T$ are transmitted and received symbol vector respectively. $n=[n_1, n_2, \ldots, n_{N_r}]$ is additive white Gaussian noise vector with variance $\sigma^2$. H is $N_t \times N_r$ MIMO channel matrix. For 2 ×2 MIMO channel:

$$\begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} + \begin{bmatrix} n1 \\ n2 \end{bmatrix} \tag{12}$$

Where $h_{11}$,$h_{12}$,$h_{21}$ and $h_{22}$ are random independent coefficients of channel matrix that are generated from Gaussian random distribution and needed to be estimated at the receiver side by using channel estimation techniques[7].

There are different MIMO detections techniques that are used to detect the original data from the received signal in MIMO system. One of the simplest methods studied is minimum mean square error (MMSE) linear detector. The output complex estimated signal of MMSE detector can be expressed as [9]:

$$\tilde{x}_{MMSE} = (H^H H + \sigma^2 I)^{-1} H^H y \tag{13}$$

where $(.)^H$ is Hermitian transpose and I is identity matrix of size $N_t \times N_r$ .

## 4. Mimo-Ofdm Implementation Using Xilinx System Generator

The block diagram of the proposed designed 2×2 MIMO-OFDM transceiver system is shown in Fig. 7. Each block is built using Xilinx system generator (XSG) tools with master clock period 10 ns. The parameters used for MIMO-OFDM system are summarized in Table 2.

In transmitter side the random bit generator, with rate 100 Mbps, is scrambled using chaos based stream ciphering method. The Convolution encoder encodes the binary scrambled stream bits with code rate ½, the output rate becomes 200 Mbps, and then matrix interleaver is used to produce interleaved stream bits. 16-ADQAM modulation is used and produces complex modulated signals with symbol rate 50 M samples/s. The signal then converted from serial to 2 parallel samples to get rate about 100 M samples /s. The signal after will be passed through the Inverse Fast Fourier Transform (IFFT) block to produce the OFDM signal, here we ignore adding cyclic prefix guard. The first signal is send in one antenna and the other in the second antenna. This operation increases the capacity of the system by two to become100 M sample/sec. After that the signals are corrupted with MIMO channel with flat fading channel and AWGN

channels. In the receiving side linear MMSE MIMO detection is used to recover the clean signals. Here we assume that the channel is perfected estimation therefore the parameters of the channels are directed entered to the MMSE MIMO detection. All the operations in the transmitted side are inversed in the received side to reconstruct the binary stream bits. The detail descriptions of each block are illustrated in the next subsections.

Table 2. MIMO-OFDM System Parameters

| Clock | Antenna configuration | Modulation | Number of subcarrier | Coding | Coding rate | MIMO processing | MIMO Signaling |
|---|---|---|---|---|---|---|---|
| 100 MHz | 2×2 | OFDM with 16-ADQAM | 16 | Binary convolutional code | 1/2 | MMSE | Spatial Multiplexing |

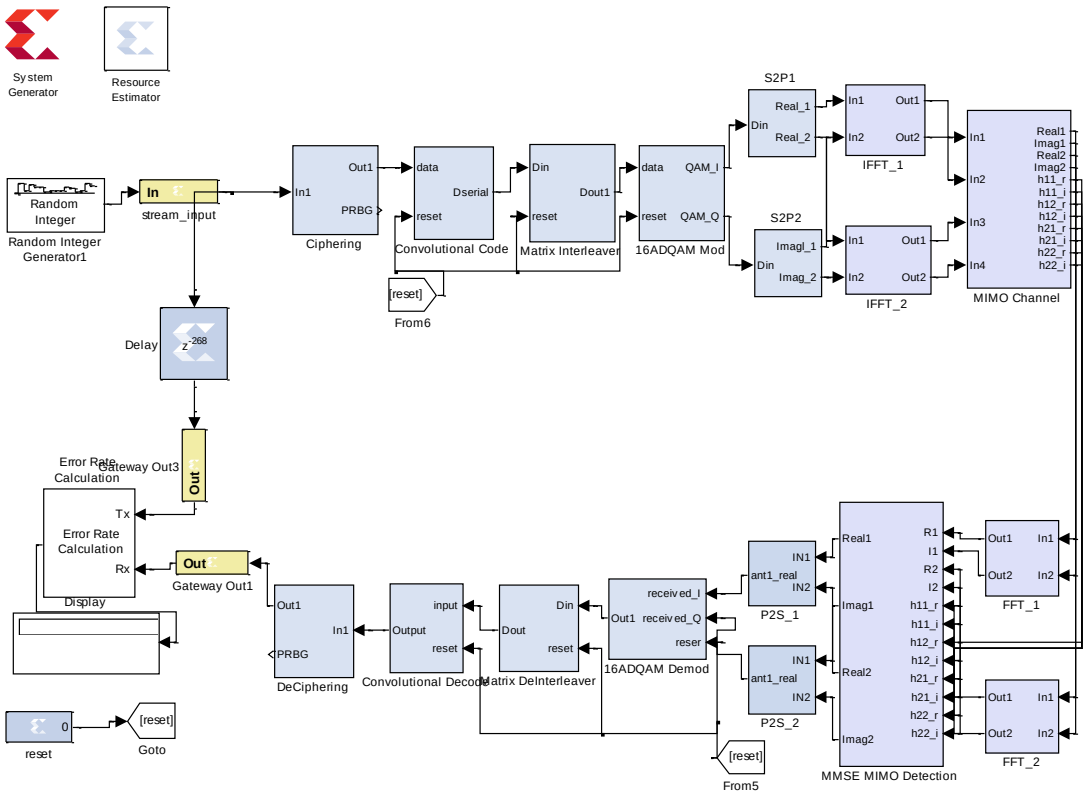



Figure 7. XSG block diagram of MIMO-OFDM transceiver.

## 4.1 Transmitter Section

### *4.1.1 Random Stream Bits generator*

Random integer generator block in Simulink is used to generate random stream bits with Bernoulli distribution function. The data rate is 100 M bps.

### *4.1.2 Data Ciphering Using Chaos*

XSG block diagram of ciphering technique is shown in Fig. 8.The format operation is 24 fixed point precision with 16 binary points.

Two chaotic logistic maps are implemented from Equation 1 and then compared using relational block to obtain the PRBG that is used as key stream bits.

In chaotic logistic map generator, the single pulse is on only for one cycle to select the initial condition. In the remaining time the feedback signal is selected in the multiplexer.

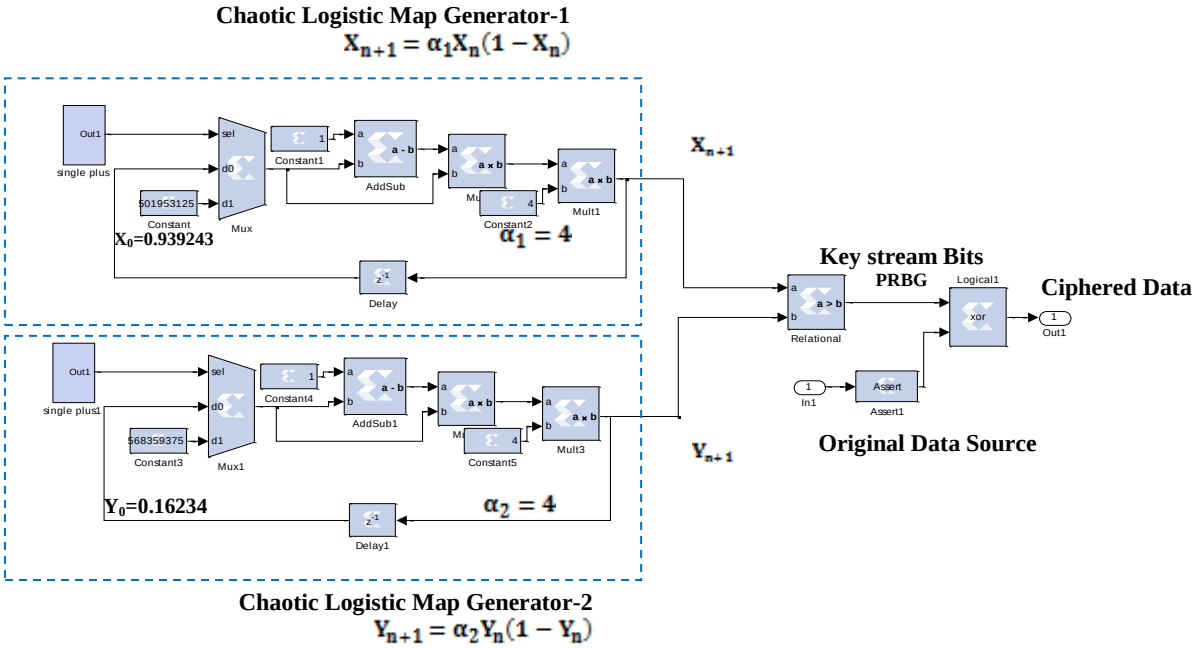



Figure 8. XSG block diagram of ciphering mode.

### *4.1.3 Convolutional Encoder*

The convolutional encoder 7.0 is Xilinx communication blockset that is IP core implemented in XSG. Fig. 9 shows the convolutional code with the following parameters: code rate=1/2, constraint length=7 and the code generator [133 171] in octal form. The output of convolutional encoder is then converted to stream bits by using parallel to serial converter.

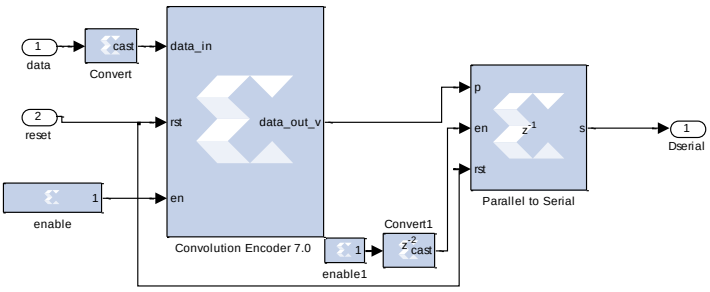



Figure 9. XSG block diagram of convolutional code.

### *4.1.4 Block Interleaver*

Interleaving is effective way to delete the effect of burst errors due to channel noise. It works by spread out the burst errors to be appeared as random at the decoder. One of the most used type for interleaving is block interleaver.

A block interleaver is implemented by writing the bits into a matrix row by row and then reading them column by column [17]. Fig. 10 shows XSG block diagram of block interleaver for 4×4 matrix.

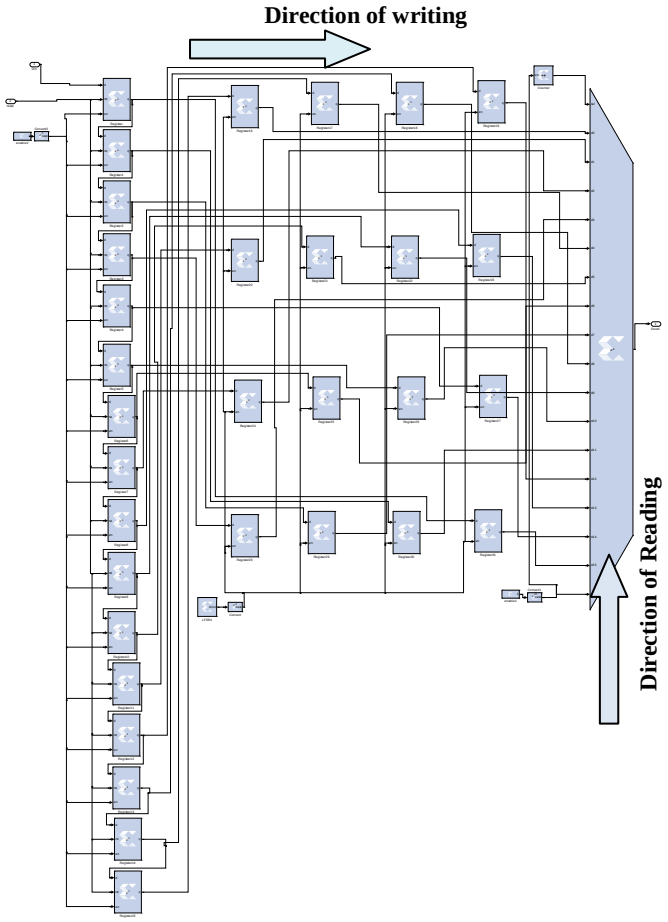



Figure 10. XSG block diagram of block interleaver.

### *4.1.5 16-ADQAM Modulator*

XSG block diagram of 16-ADQAM modulation is shown in Fig. 11. The stream interleaved data is converted to parallel four bits $\{b_3b_2b_1b_0\}$ by using serial to parallel blockset. Each two bit is concatenated to unsigned value using concatenate blockset and used as addressing in RAM memory to select one of the following phase { 0 ,1.5708, -1.5708, 3.141} with fixed point of word length=13 and fractional bits length=10, this is suitable format for this function.

After that sine and cosine functions of the differential angle ($\Delta\theta$) are calculated using CORDIC SINCOS blockset to produce $e^{j\Delta\theta}$. Then, $e^{j\Delta\theta_1}$ and $e^{j\Delta\theta_2}$ are multiplied by $C_r(i-1)$ and $D_r(i-1)$ respectively using complex multiplication (CM) block to produce $C_r(i)$ and $D_r(i)$ as in (3) and (4) respectively. Complex multiplication block is shown in

Fig. 12. Finally, the complex s(i) can be calculated using AddSub blockset between $C_r(i)$ and $D_r(i)$ according to (2).

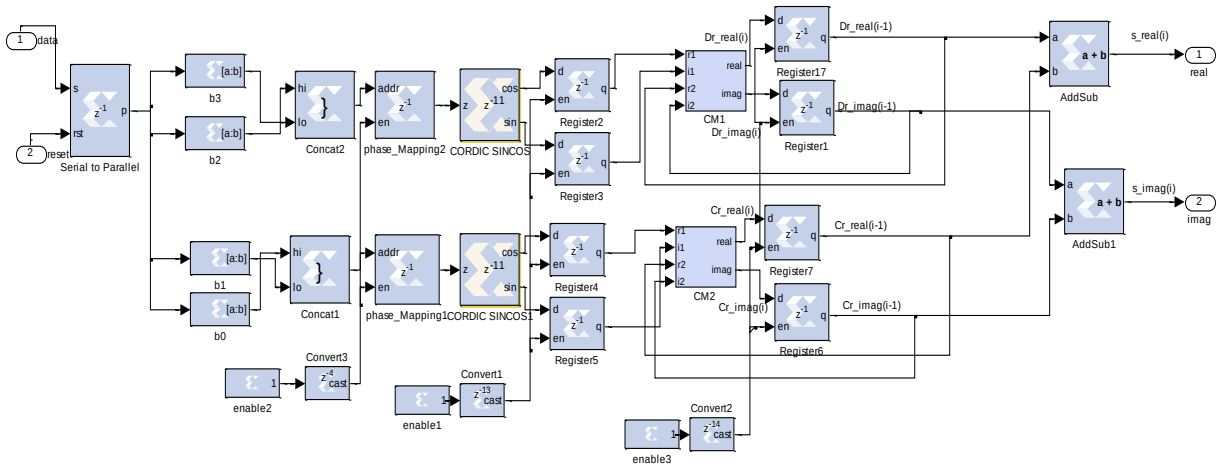



Figure 11. XSG block diagram of 16-ADQAM modulation.

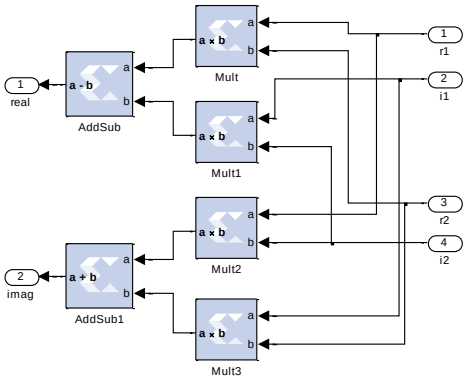



Figure 12. XSG block diagram of two complex number multiplication.

### *4.1.6 Serial to parallel Converter*

The serial 16-ADQAM symbols are converted from serial symbols to two parallel symbols using serial to parallel block shown in Fig. 13.This produce two symbols one to be used for the first channel and the other for second channel to produce two transmits signal. We need two block of serial to parallel converter one for real and the other for imaginary part.

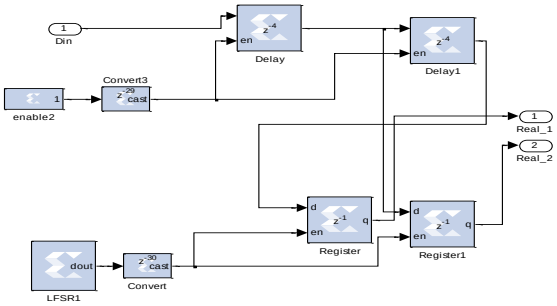



Figure 13. XSG block diagram of serial to parallel converter.

### *4.1.7 Inverse Fast Fourier Transform (Ifft)*

In the first the signal is converted to parallel 16 samples as shown in Fig. 14. The XSG block diagram of serial to parallel converter of 16 samples is shown in Fig. 15. The complex parallel 16 samples are then ready to enter into IFFT to obtain OFDM signal. IFFT converts the ADQAM signal from frequency to time domain. These 32 parallel complex symbols split into two frame each of 16 complex symbols. The first frame entered the first IFFT block to produce the first OFDM complex signal and the second 16 complex symbols are entered to the second IFFT block to produce the second OFDM complex signal. Fig. 16 shows XSG block diagram of IFFT. In the first the 16 complex signals are shuffled according to bit reversal order. After that the shuffled data is passing through Stage_1, Multiple_1, Stage_2, Multiple_2, Stage_3, Multiple_3 and Stage_4 modules that it are implemented using XSG as shown in Figs. (17-23) respectively. Each format operation is fixed points with 20 word length bit and 16 fractional bits. This format is optimum choose for IFFT when compared with that of Matlab values we obtain about 0.005 mean square error.

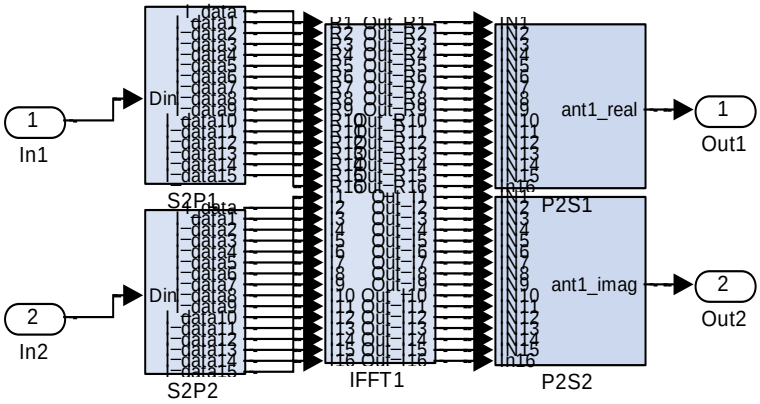



Figure 14. XSG block diagram of IFFT.

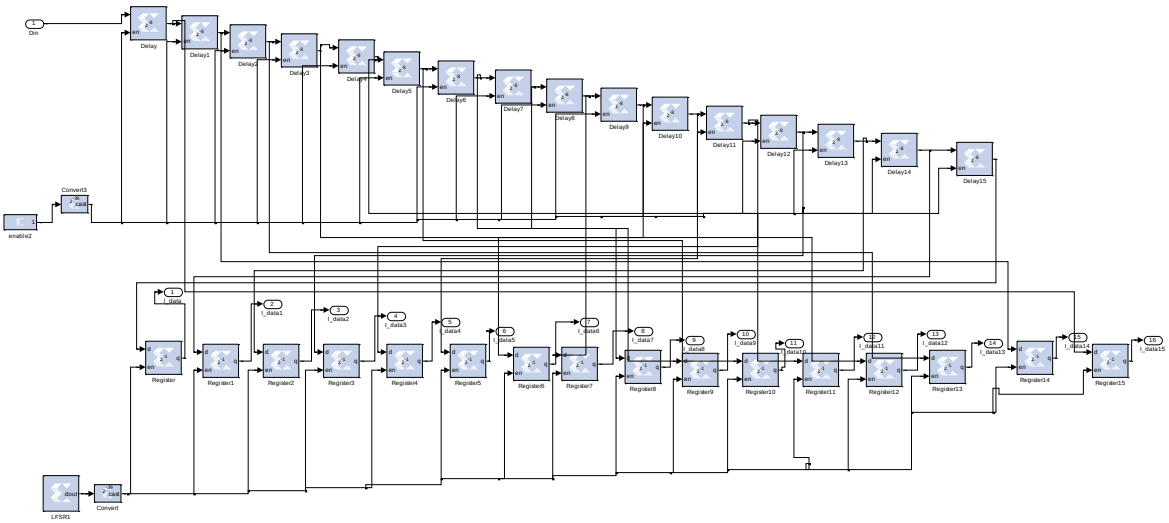


Figure 15. XSG block diagram of serial to parallel 16 samples

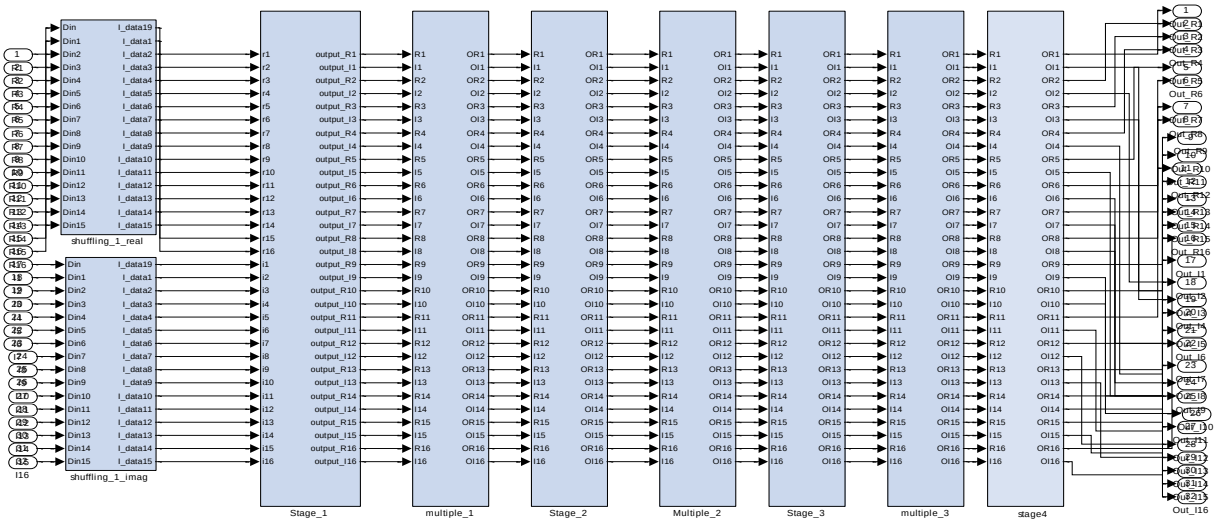



Figure 16. XSG block diagram of 16 points decimation in time IFFT.

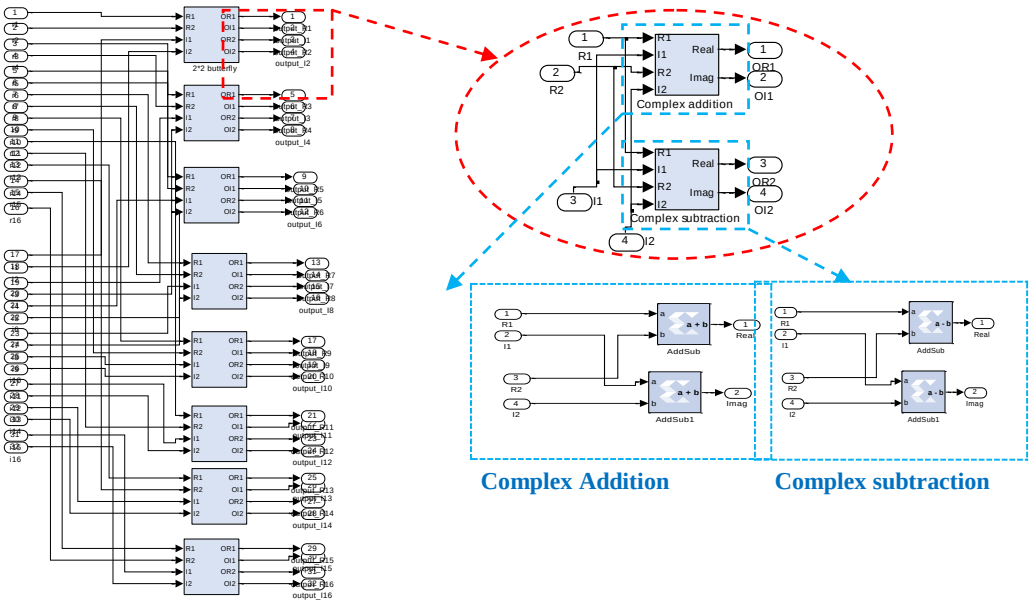



Figure 17. Stage_1 Module of IFFT.

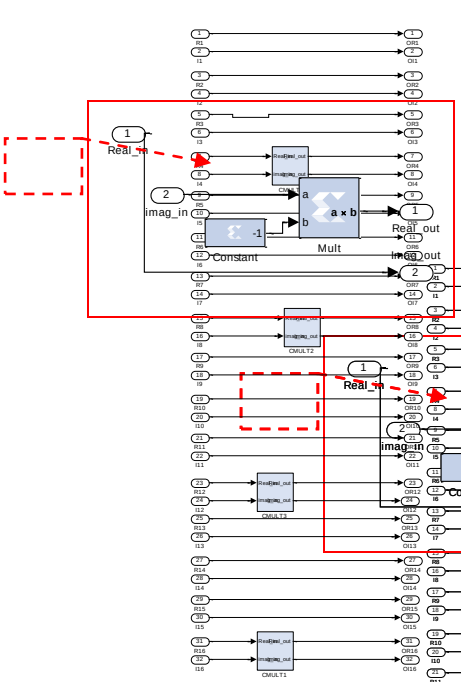



Figure 18. Multiple_1 Module of IFFT

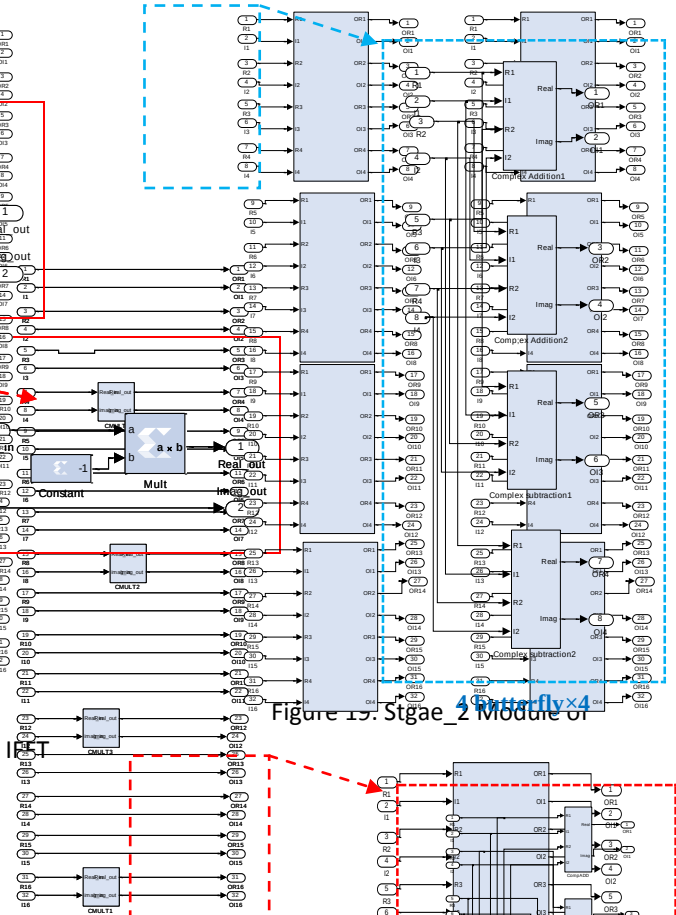



Figure 19. Stgae_2 Module of IFFT

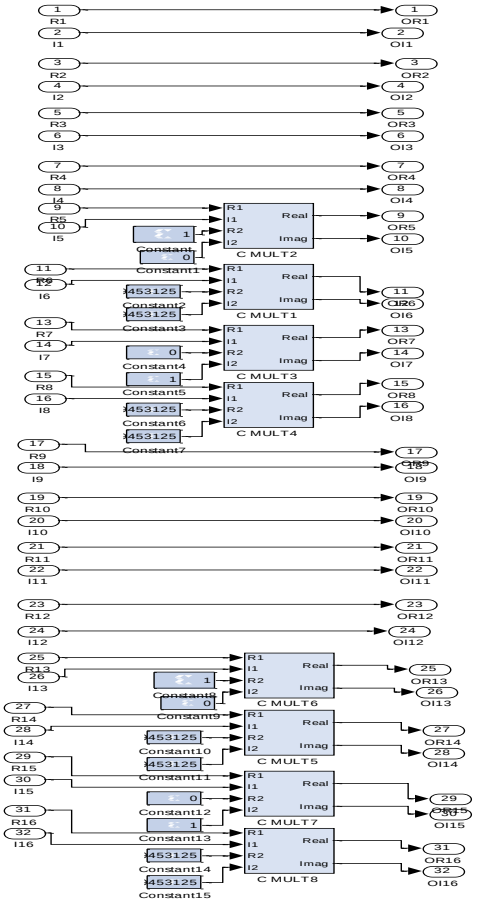



Figure 20. Multiple_2 Module of IFFT

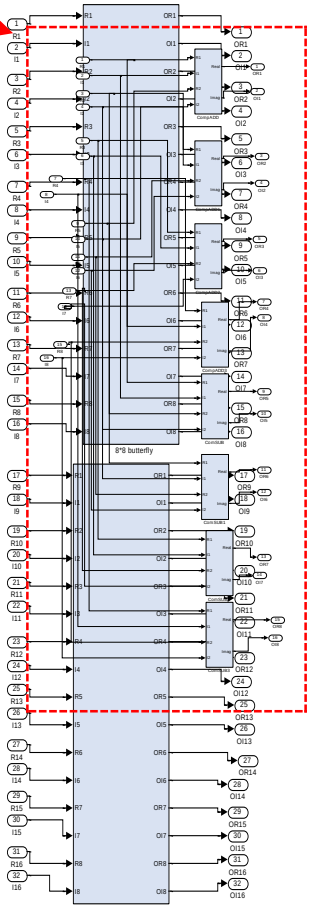



Figure 21. Stgae_3 Module of IFFT

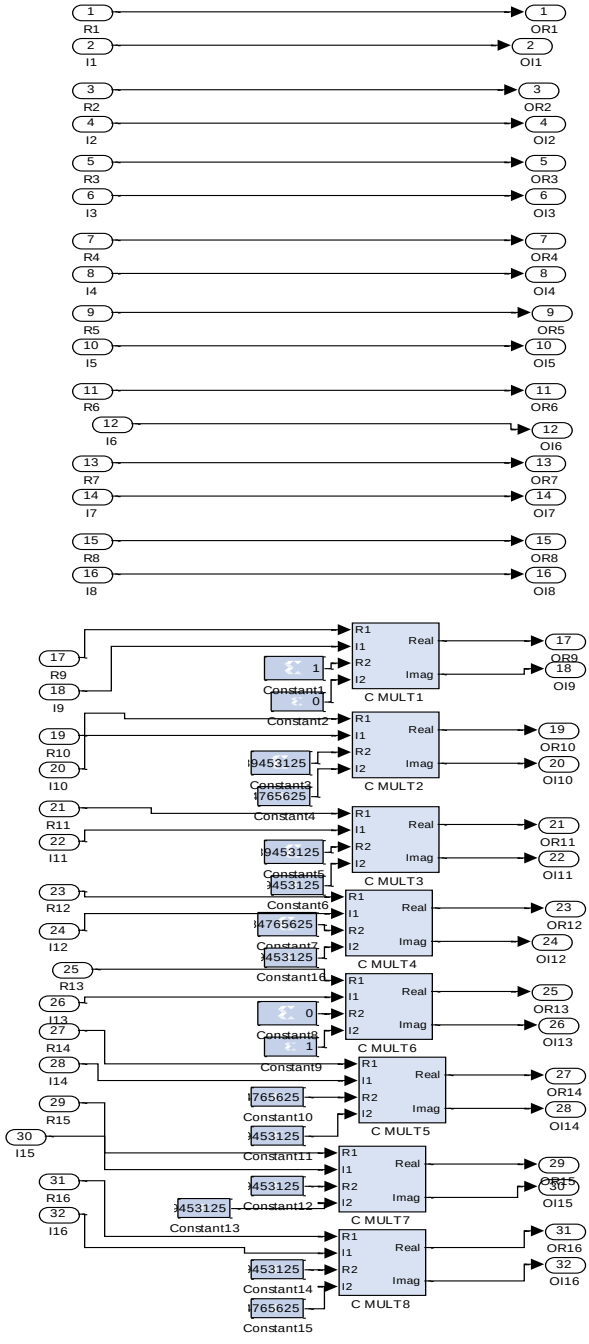



Figure 22. Multiple_3 Module of IFFT

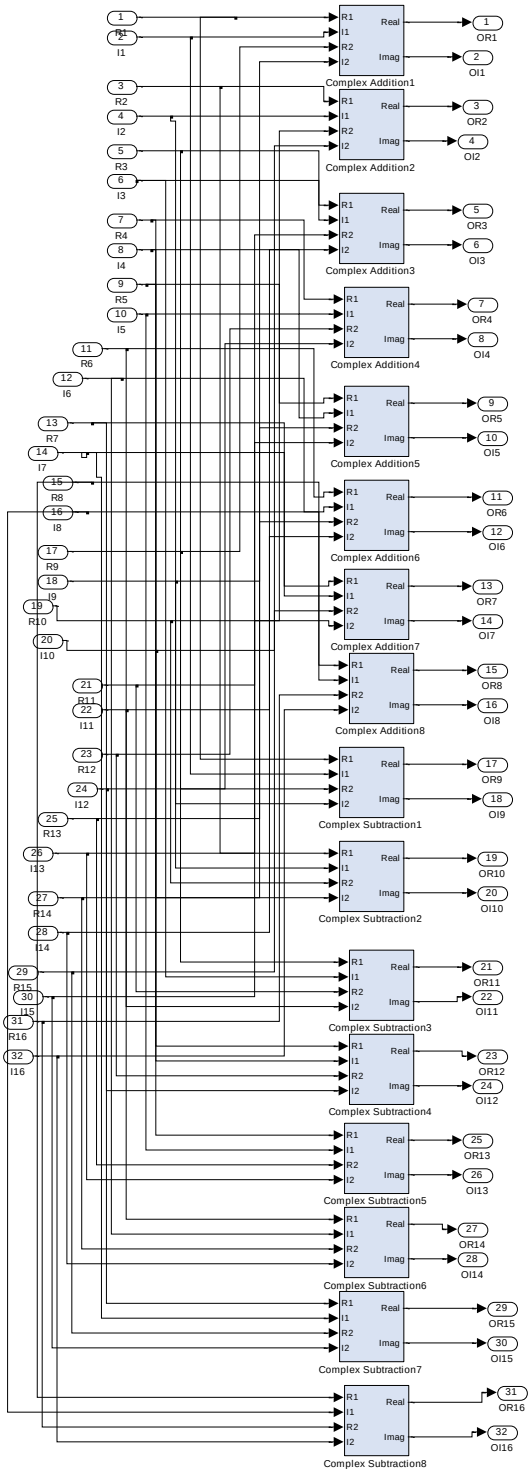



Figure 23. Stgae_4 Module of IFFT

The 16 parallel OFDM signal is then converted to serial symbols again using parallel to serial converter. It is required two converters for each OFDM signal one for real and the other for imaginary part. Fig. 24 shows XSG block diagram for one of parallel to serial converter. These two serial complex OFDM signals are spatially multiplexed over MIMO channel.

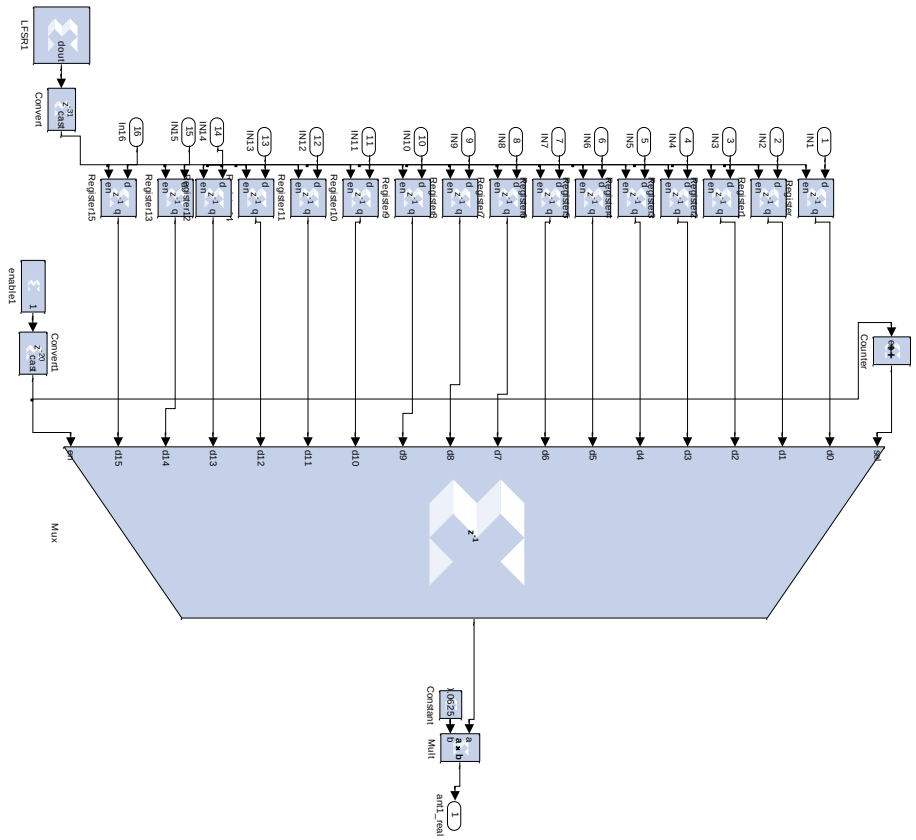


Figure 24. XSG block diagram of parallel to serial converter

### *4.1.8 MIMO Channel*

Fig. 25 shows XSG block diagram of 2×2 MIMO flat fading with AWGN channel. Each element in H matrix {$h_{11}$,$h_{12}$,$h_{21}$ and $h_{22}$} is implemented using white Gaussian noise generator blockset in communication Xilinx reference blockset.

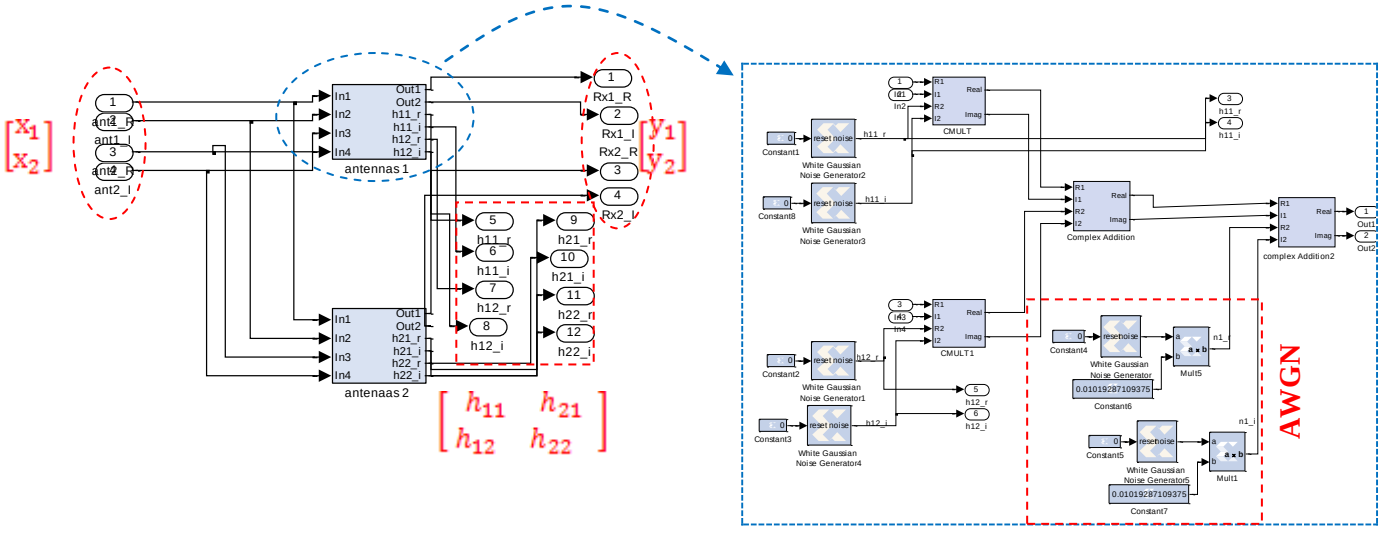



Figure 25. XSG MIMO block diagram of flat fading with AWGN channel.

## 4.2 Receiver Section

### *4.2.1 Fast Fourier Transform (FFT)*

In the first**,** the received two complex signals are converted into 16 parallel samples. It is required two converters for each received signal one for real and the other for imaginary part. The total 4 serial to 16 parallel converters required. Secondly, the parallel samples are entered into FFT to recover the original modulating signal and after that re-back to serial sample. Fig. 26 shows XSG block diagram of FFT. XSG block of FFT is identical to IFFT block only difference the twiddle factor becomes conjugate. Also serial to parallel and parallel to serial is identical to Figs. 15 and 25 respectively only difference is setting times.

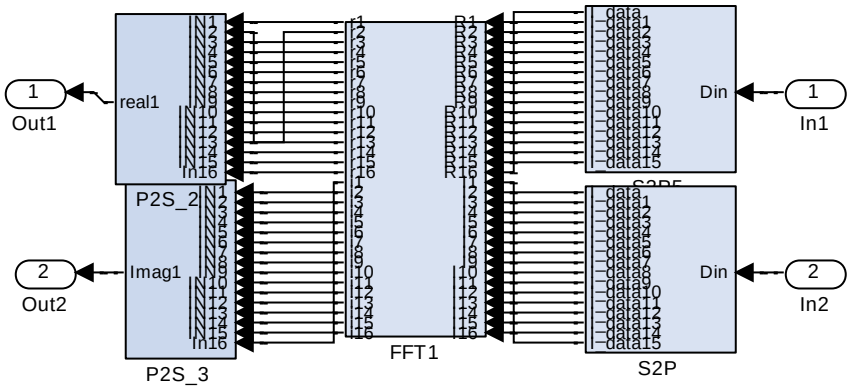



Figure 26. XSG block diagram of FFT.

### *4.2.2 MMSE Signal Detection*

Fig. 27 shows the XSG block diagram of MMSE 2×2 MIMO detection. We assume here the channel matrix is perfect estimated, therefore the coefficients of H matrix is directly entered into the MIMO detection.

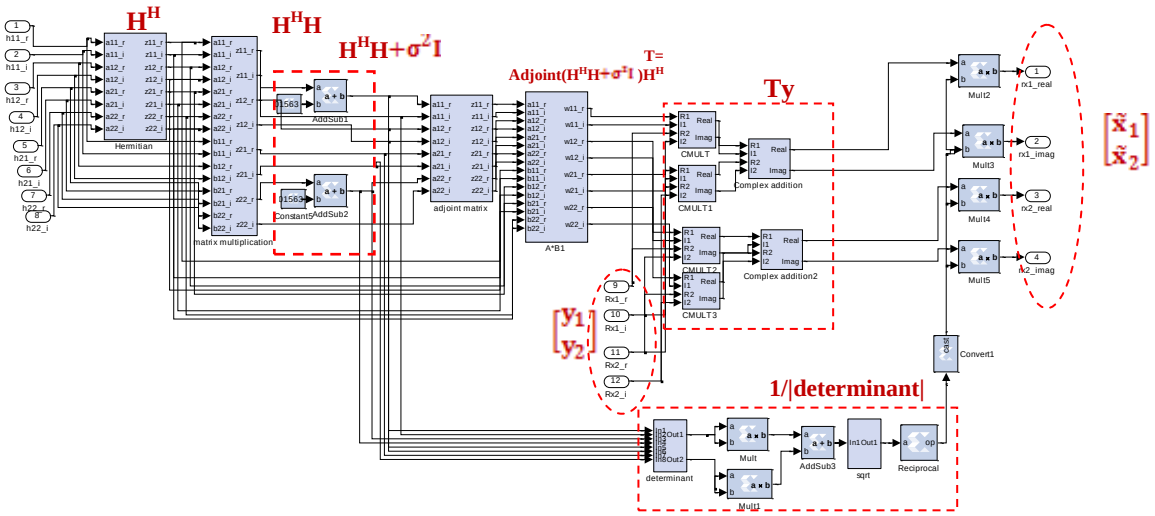



Figure 27. XSG block diagram of MMSE detector.

### *4.2.3 16-ADQAM Demodulator*

XSG block diagram of 16-ADQAM demodulation is shown in Fig. 28. This system is implemented according to (7-10). Fig. 29 shows XSG block of bit detection that is implemented according to (9) and (10). The precision format of $\hat{C}_r$ and $\hat{D}_r$ are 20 fixed points with 16 fractional bits. Cast blockset needs to convert 20 fixed points precision to 6 fixed points with 0 fractional bits.

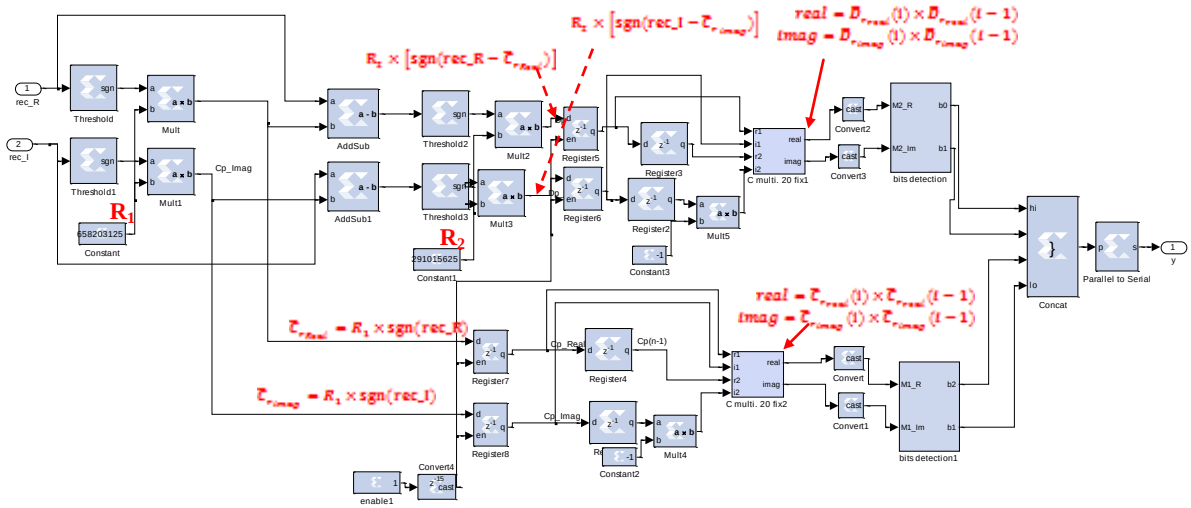
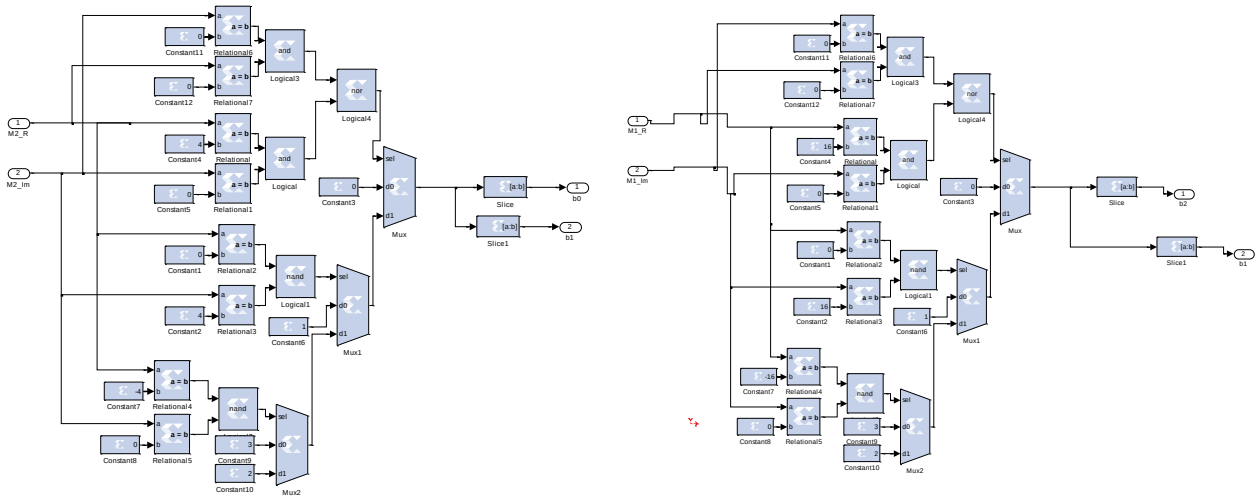



Figure 28. XSG block diagram of 16-ADQAM demodulation.




a) Bit Detection ($b_0$ $b_1$) b) Bit Detection ($b_2$ $b_3$)

Figure 29. XSG block diagram of bit detection module.

### *4.2.4 Block Deinterleaver*

The same block interleaver can be used to deinterleaver and recover the original stream bits and its addressing.

### 4.2.5 Viterbi Decoder

Fig. 30 shows XSG block diagram of Viterbi decoder. In the first the serial bits it is converted to parallel 2 bits and then using Viterbi decoder version 7.01 Xilinx blockset to decode the signal. The parameters of the block are constrain length=7, hard coding, output rate=2 and code generator [133 171] in octal form.

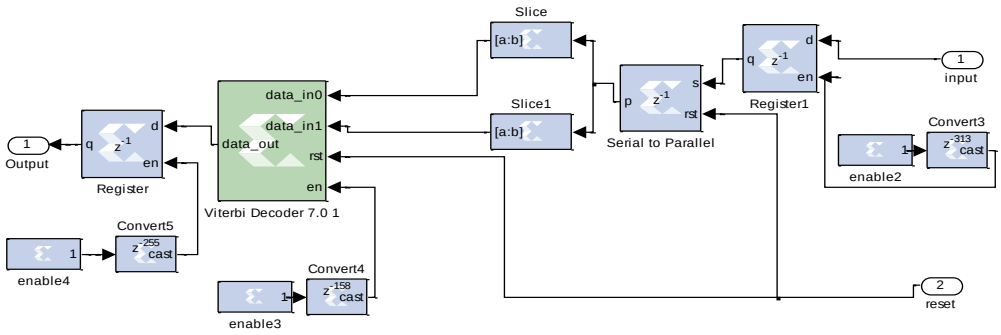



Figure 30. XSG block diagram of Viterbi decoder.

### 4.2.6 Descrambling Using Chaos

The block diagram of deciphering is identical to that at transmitter the only different is the pseudo random bit generator followed by delay components. This delay component must be found by time synchronization between the transmitter and the receiver. This work does not exist here.

## 5. Matlab Simulation Results

The MIMO-OFDM system is simulated using Matlab 7.12 with the following parameters: 16 subcarrier numbers, convolutional code (7 constraint length, ½ code rate and $[133\ 171]_{octal}$ generator polynomial) and 2×2 MMSE MIMO detection. Fig. 31 shows the performance comparison of MIMO-OFDM under flat fading channel with and without channel coding. We assume in this simulation the channel is perfect estimated. It is seen that from this figure at BER=$10^{-3}$, there is 7 dB gain can be added by using convolutional code.

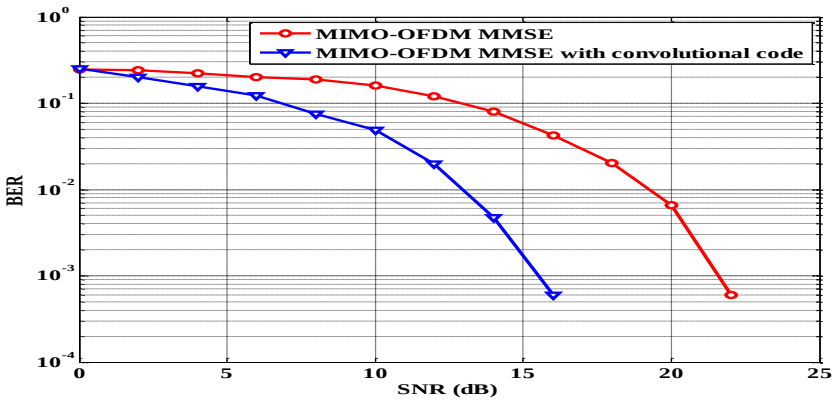



Figure 31. Performance of MIMO-OFDM system over flat fading channel.

## 6. Xsg Simulation Results

The Resulting waveforms that explain the outputs of each component in MMSE MIMO-OFDM system that are implemented using XSG are plotted using MATLAB program. Fig. 32 shows XSG simulation waveforms of ciphering system. $X_n$ and $Y_n$ are chaos signals used to generate the key signal, PRBG where is adding mod 2 with original data source to produce ciphered message. The bit rate is remaining the same.

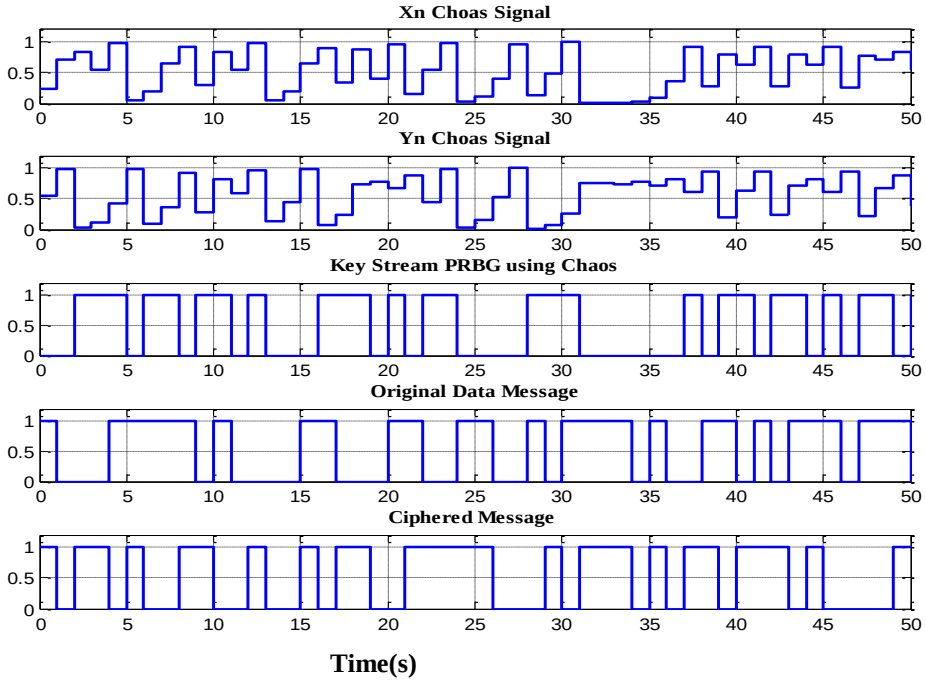



Figure 32. XSG simulation waveforms of the ciphering system.

Fig. 33 shows XSG simulation waveforms of convolutional code. It can be seen from this figure the rate is increased by two to become 200 Mbps. Fig. 34 Shows Experimental signals of Matrix Interleaver for 4×4 matrix. The control signal is used to start loading the signals to registers to become serial at the output of multiplexer. This signal becomes on every 16 bits cycle.

The output rate remains the same. Fig. 35 shows XSG simulation waveforms of 16-ADQAM modulation. As can be seen from this figure the rate at the output of serial to parallel converter is 50 M samples/sec (20 ns).

The 16-ADQAM signal is parsing into two signals one for each channels to become the rate 100 M samples/sec (40 ns). Fig. 36 shows the OFDM transmitted complex signals the received complex signal and the MMSE detection complex signal of the first channel. Fig. 37 shows the comparison results between transmitted and received signals.

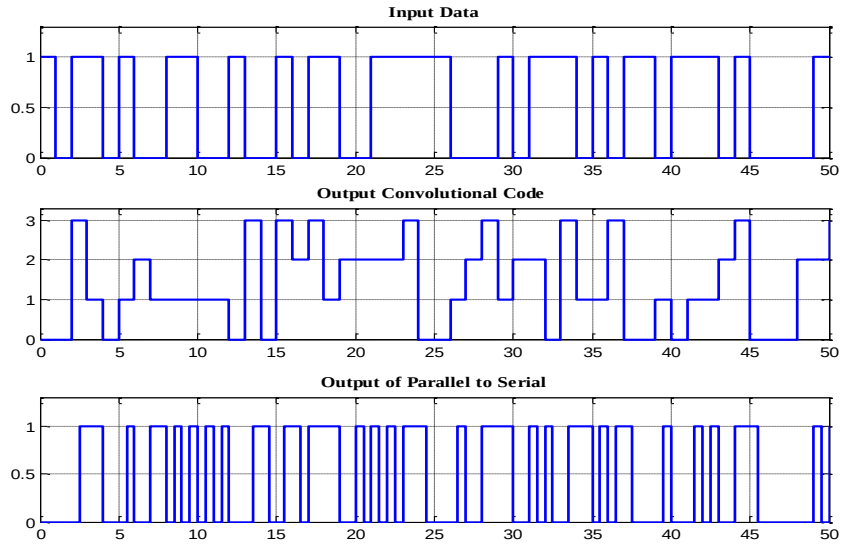



Figure 33. XSG simulation waveforms of convolutional code.

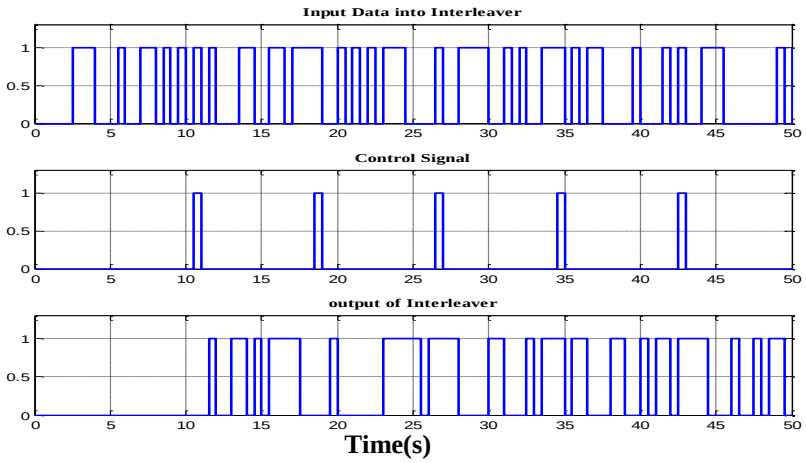



Figure 34. XSG simulation waveforms of Matrix Interleaver for 4×4 matrix.

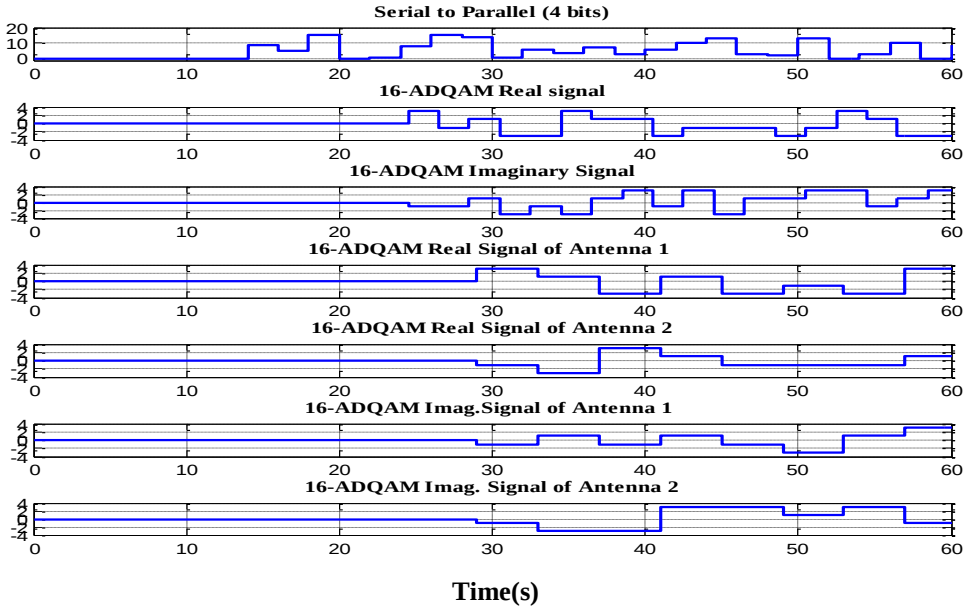



Figure 35. XSG simulation waveforms of 16-ADQAM modulation.

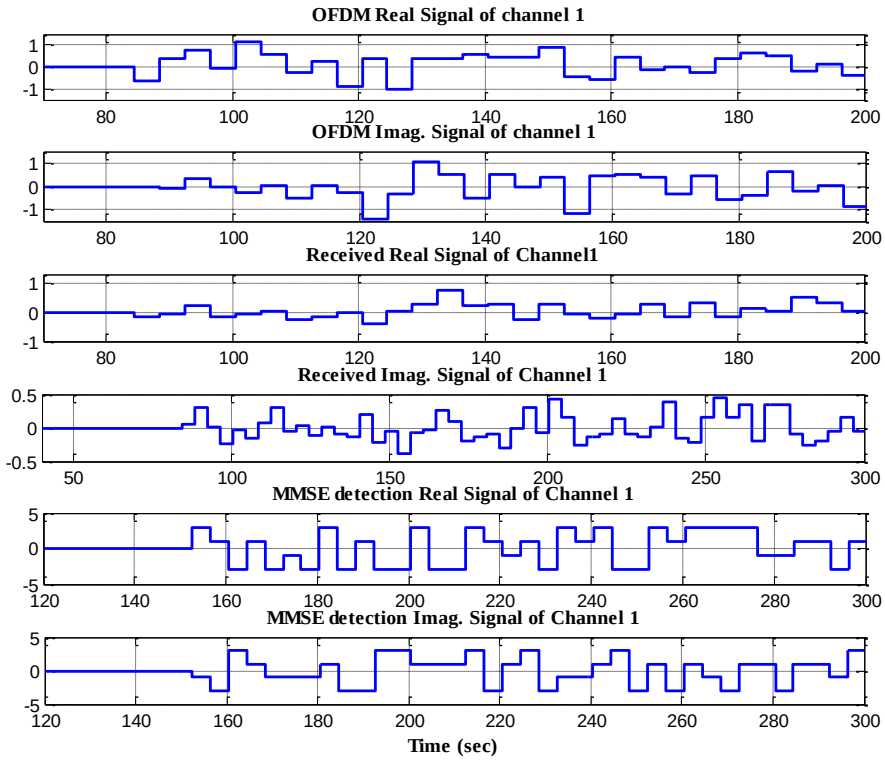



Figure 36. XSG simulation waveforms of OFDM and MMSE detection.

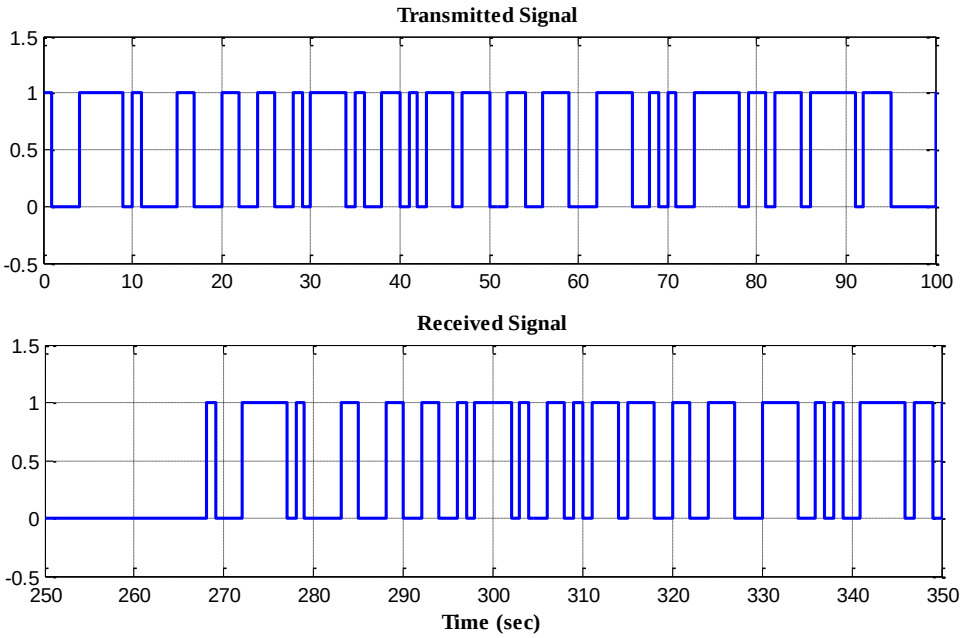



Figure 37. Comparing transmitted and received data.

## 7. Synthesis Report

After completing system design and simulation results, VHDL Code or Bitstream file is generated using select device which is virtex 4. Fig. 38 shows the parameters of

system generator. Resources used for the system are found by using ISE 14.1 program as shown in Table 3.

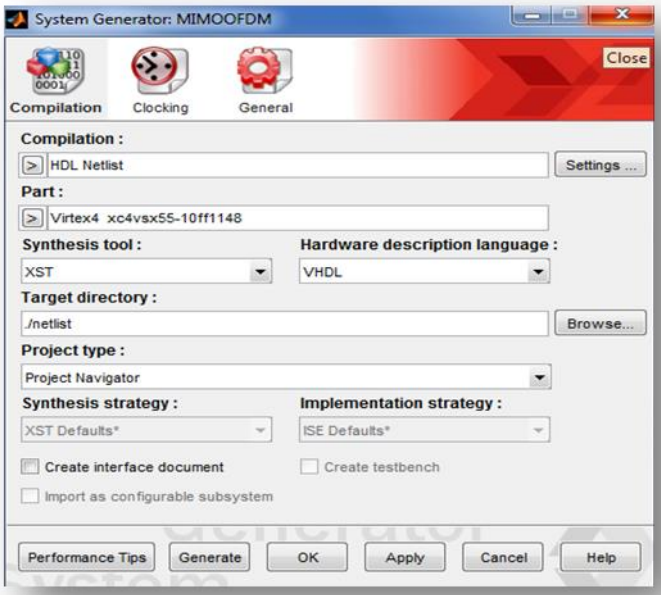



Figure 38. System generator parameters.

Table 3. Development system resources

| Device Utilization Summary | | | |
|---|---|---|---|
| **Logic Utilization** | **Used** | **Available** | **Utilization** |
| Number of Slice Flip Flops | 10,361 | 49,152 | 21% |
| Number of 4 input LUTs | 28,128 | 49,152 | 57% |
| Number of occupied Slices | 17,463 | 24,576 | 71% |
| Number of Slices containing only related logic | 17,463 | 17,436 | 100% |
| Total Number of 4 input LUTs | 31,365 | 49,152 | 63% |
| Number of bonded IOBs | 15 | 640 | 2% |
| Number of BUFG/BUFGCTRLs | 1 | 32 | 3% |
| Number of FIFO 16/RAMB 16s | 4 | 320 | 1% |
| Number of DSP 48s | 450 | 512 | 87% |

## 8. Conclusions

In this paper, 2×2 MMSE MIMO-OFDM system is implemented using Xilinx system generator with the following adding features: increasing security of the system using chaos stream ciphering, using 16-ADQAM modulation to solve the ambiguity problem and implement FFT in pipelining way. The results show that the signal is recovered in correctly at the receiver side and the VHDL code file can be generated in successful when Xilinx Virtex-4 device is used.

## 9. Reference


1. G.V Ganesh, B .Murali Krishna, K. Sravan Kumar, Prathyusha, R.Venkatesh T.Vineel Jessy (2015). FPGA Implementation Of OFDM Transmitter Using Simulink and Xilinx System Generator. Journal of Theoretical and Applied Information Technology, Vol.78. No.1, pp.125-131.
2. Li- Wen Hsu and Dah chung chang (2014). 2×2 STBC MIMO-OFDM Receiver Design for WLAN with estimation of different carrier frequency offset. International Journal of Electronic Engineering and Telecommunication (IJEETC), Vol.3, No.1, pp. 1-13.
3. Amar Mandekar, S. D. Patil, D. S. Arora (2014). Implementation of 802.11n MIMO-OFDM Transceiver. International Journal of Scientific and Engineering Research, Vol. 5, Issue 7.
4. Johanna Kerttula, Markus Myllylä, Markku Juntti (2007). "Implementation Of A K-Best Based MIMO-OFDM Detector Algorithm". European Signal Processing Conference (EUSIPCO), Poznan, pp. 2149-2153.
5. Shingo Yoshizawa1 and Yoshikazu Miyanaga (2012). Hardware Development of Baseband Transceiver and FPGA-Based Testbed in 8×8 and 2×2 MIMO-OFDM Systems. ECTI Transactions on Computer And Information Technology, Vol.6, No.1 pp. 37-43.
6. Jeoong S. Park, Hong-Jip Jung and Viktor K. Prasanna (2012). Efficient FPGA-based Implementations of the MIMO-OFDM Physical Layer. Circuits Systems and Signal Processing, Vol. 31, Issue 4, pp. 1487-1511.
7. Jesse Chen,Weijun Zhu, Babak Daneshrad, Jatin Bhatia, Hun-Seok Kim, Karim Mohammed, Sandeep Sasi, Anish Shah (2007). "A Real Time 4x4 Mimo-Ofdm Sdr For Wireless Networking Research". European Signal Processing Conference (EUSIPCO), Poznan, Poland, pp. 1151-1155.
8. R . Rakesh and Ramesh (2014). FPGA Implementation of Channel Estimation Technique in MIMO-OFDM System. International Journal of Engineering Research & Technology (IJERT), Vol. 3 Issue 5, pp. 1691-1694.
9. Rima Raissawinda, Gede Puja, Yoedy Moegiharto, Ahmed Zainudin and Imam Dui Augsalim (2014). Channel Estimation Design of MIMO-OFDM Systems Using MMSE for IEEE 802.11n WLAN Standard. European Scientific Journal, Vol. 10, N0.6, pp.137-145.
10. Mahendra Babu D.S., Vinutha M.R and Uma C. (2014). Design and Implementation of MIMO-OFDM using Encoding and Decoding techniques on FPGA. International Journal of Scientific & Engineering Research, Volume 5, Issue 6, pp. 939-944.
11. K.Sathya,(M.E), S.S.Raju, M.E. ,V. Prabu Kumar, M.E. , T.Shiva Prasad,M.E. (2013). Design of MIMO-OFDM Using Fixed Sphere Decoding Detection Method. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), Volume 3, Issue 6, pp. 63-66.
12. A.Vijaya Bhaskar And C.Ravi Shankar Hanuman (2012). Zuc Stream Cipher Using Feedback Carry Shift Register. International Journal Of Engineering Science And Technology (IJEST), Vol. 4, No.06, pp. 2462-2467.

13. Saad Nahim Al Saad and Eman Hato (2014). A speech Encyption based on Chaotic Maps. International Journal of Computer Applications, Vol. 93, No.4, pp. 19-28.
14. Vinod Patidar, K. K. Sud and N. K. Pareek (2004). A Pseudo Random Bit Genertor based on Chaotic Logistic Map and its Statistical Testing. Informatica, Vol. 33, pp.441-452.
15. Jeng-Kuang Hwang and Yu-Lun Chiu (2008). "Performance Analysis of an Angle Differential-QAM Scheme for Resolving Phase Ambiguity". Advanced Communication Technology, 10th International Conference, Vol. 1,pp.161-166.
16. Aruna Arya, Augusta Sophy (2013). FPGA Implementation of 32 point Radix-2 Pipelined FFT. International Journal of Research in Electronics & Communication Technology, Vol. 1, Issue 1, pp. 72-77.
17. Renu Panchal and Rahul Gedam (2012). VHDL Implementation of Reconfigurable Multimode Block Interleaver for OFDM Based WLAN. International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE), Vol. 1, Issue 4, pp. 81-85.
18. Release 10.1 (2008). "System Generator for DSP: Getting Started Guide".
19. John G. Proakis and Dimitris G. Manolakis (2007), "Digital Signal Processing: Principles, Algorithms and Applications". Fourth Edition, Pearson Prentice Hall.