FPGA Implementation Platform for MIMO-OFDM Based on UART

Sherif Moussa, Ahmed M. Abdel Razik, Adel Omar Dahmane, Habib Hamam

Elec and Comp. Eng. Department, Université du Québec à Trois-Rivières, Trois-Rivières, Canada
School of Engineering, Canadian University of Dubai, Dubai, UAE
School of Eng., Arab Academy for Science, Technology and Maritime Transport, Cairo, Egypt
Faculty of Engineering, University of Moncton, Moncton, Canada

Abstract

Most hardware designers use Simulink as a design platform because it contains many components that have hardware equivalent. In addition to that, it supports Matlab Co-simulation and hardware synthesis. Simulink is seen as a good platform for low complexity systems that do not require trimming analysis during the verification phase. Hardware design for complex systems such as MIMO-OFDM needs an immediate manual conversion from Matlab to RTL-VHDL, and the communication between Matlab and FPGA must be managed directly through the Universal Asynchronous Receive and Transmit (UART). This paper proposes an integrated architecture for a UART module to be used with MIMO-OFDM hardware platform, the purpose of this module is to enable the communication between Matlab and FPGA board. It consumes nearly 1% from the overall resources of the target FPGA.

Keywords: MIMO, OFDM, UART, FPGA

1. INTRODUCTION

Orthogonal Frequency Division Multiplexing (OFDM) is a promising technology for high data rate wireless communications due to its robustness to frequency selective fading, high spectral efficiency, and low computational complexity. Multiple Input Multiple Output (MIMO) systems, which use multiple transmit and receive antennas, are often used with OFDM to improve the channel capacity [1][2].

FPGAs are suitable for computationally intensive arithmetic calculations like matrix inversion as they provide well-suited architectural features such as large number of programmable logic elements, distributed ram, block ram, DSP slices, register slices and look up tables (LUTs) in addition to the IP cores offered by the FPGA vendors which facilitates the design process and reduce the design time.

To design and implement MIMO-OFDM transmitting and receiving systems on FPGA, the process shown in Figure 1 starts by identifying the characteristics of the transmitted signal, the status of the transmission channel and the received signal model taking into account the anticipated signal impairments. This stage ends by developing a high-level simulation model for the system on Matlab [3].

After completing the Matlab model successfully, the complete Matlab design is translated to a hardware design using VHDL code and IP cores, using a Register Transfere level (RTL) design approach. Then Matlab/VHDL co-simulation is carried out to verify that the hardware design is performing exactly the required function. This step is associated with performing several advanced design optimization techniques including design for timing performance, pipeline techniques and designing for area optimizations and resource sharing.

Fig 1: Design methodology flow chart

After the RTL design and optimizations, the post place & route simulations are carried out to make sure that the optimized design is meeting the design constraints. Finally the on board verification is required to ensure that the hardware design is performing the expected function and producing the expected results.
Fig 2: Simplified block diagram of MIMO-OFDM system

During this step, the design data path needs to communicate with sources and sinks of the user data in order to perform co-simulation with Matlab code. Matlab is used to set the simulation parameters such as modulation scheme, algorithm choice, input data, and so forth; it is also used to start and stop the simulation. Most of the reported design in the literature [4][5] use Simulink as hardware design platform, because it provides the designer with a library of components which have a hardware equivalent. However, Simulink is suitable for systems which are not too complex because it is preferred for DSP calculations, not for systems with sophisticated control. MIMO-OFDM system have complex control signals and it’s not easy to describe it in Simulink. Furthermore, the timing parameters need to be added to the design during the RTL development; clock is supported by Simulink by using “z^{-1}” delay blocks. However, in order to model the right number of clock cycle delays the process is considered a time consuming and error-prone task. Hence, Simulink is not suitable platform to develop cycle-true behavior for high complexity systems such as MIMO-OFDM base station.

For the above reasons, a direct, manual conversion from Matlab code to RTL-VHDL is used and the communication between Matlab and FPGA has to be managed directly through the Universal Asynchronous Receive and Transmit (UART). UART allows full-duplex communication using serial link, and it is widely used in the data communications and control system. Building the UART function using separate interface chip causes a waste of hardware resources; hence it’s better to integrate the UART function inside the same FPGA [6]. In this paper, UART core functions are implemented using VHDL and integrated with MIMO-OFDM FPGA chip to achieve compact, stable and reliable data transmission, which effectively represent a complete hardware design platform for MIMO-OFDM system.

2. MIMO-OFDM SYSTEM

A general MIMO-OFDM system is shown in Figure 2, where \( N_t \) transmit antennas, \( N_r \) receive antennas, and \( N_f \)-tone OFDM are used. First, the incoming bit stream is mapped into a number of data symbols via some modulation type such as BPSK. Then a block of \( N_s \) data symbols \( \{ s_1, s_2, \ldots, s_{N_s} \} \) are encoded into a codeword matrix \( S \) of size \( T \times N_t \), which will then be sent through \( N_t \) antennas in \( T \) OFDM frames. We can identify \( X \) as a subset of \( S \) that represents the transmitted symbols from all transmitting antennas for subcarrier \( k \), hence

\[
X = \begin{bmatrix}
S_{1k} \\
\vdots \\
S_{N_tk}
\end{bmatrix}
\]

(1)

After passing through the MIMO channels, the received signals will be first sent to the OFDM demodulator (cyclic prefix removal and FFT). Without loss of generality, and by considering one subcarrier \( k \), the received vector could be represented as

\[
Y = HX + V
\]

(2)

Where \( Y \) is the received vector with \( N_r \) dimension, \( H \) is an \( N_t \times N_r \) complex propagation matrix and \( V \) represents zero mean complex Additive White Gaussian Noise. The data symbols \( \hat{S} \) is then estimated by linear detection algorithm such as Zero Forcing and is given by

\[
\hat{S} = H^{-1}Y
\]

(3)

3. UART ALGORITHM

UART module consists of transmitter and receiver modules. The transmitter is built as a shift register that accept parallel data and then produce it serially at a specific rate. The receiver, on the other hand, shifts in data bit by bit and then produces it in parallel at the output. Figure 2 shows that the transmitter serial output is ‘1’ during the idle status. Then a start bit of ‘0’ is used to indicate the beginning of transmission, then 6, 7, or 8 data bits are sent, followed by an optional parity bit. Finally it sends ‘1’ bit to indicate the stop of data transmission. The parity bit is set to ‘0’ when the data bits have an odd number of 1’s, if odd parity is used. In case of even parity, it is set to ‘0’ if the data bits have an even number of 1’s.
2 MIMO-OFDM system.

Accordingly.

transmitted bits and then retrieve them at these points

oversampling is used to estimate the middle points of

parameters in order to retrieve the data bits. A baud rate

transmitted signal, the receiver uses the preconfigured

that the clock information is not included in the

interface modules as shown in Figure 4. Due to the fact

subsystem consists of receiver, baud rate generator, and

Fig 3: UART serial bit structure

Figure 3 shows a UART transmission system

that uses 8 data bits with no parity bit and 1 stop bit. In

this system it could be noted that the data least significant

bit is transmitted first. Since clock information is not

included through the serial line, the transmitter and

receiver must agree on the communication parameters

before transmission starts. These parameters include the

baud rate (i.e., number of bits per second), the number of

data bits and stop bits, and use of the parity bit. The

design of the receiving and transmitting subsystems is

described in the following sections. The design is

customized for a UART with a 19,200 baud rate, 8 data

bits, 1 stop bit, and no parity bit.

4. UART RECEIVER ARCHITECTURE

The architecture of the UART receiving

subsystem consists of receiver, baud rate generator, and

interface modules as shown in Figure 4. Due to the fact

that the clock information is not included in the

transmitted signal, the receiver uses the preconfigured

parameters in order to retrieve the data bits. A baud rate

oversampling is used to estimate the middle points of

transmitted bits and then retrieve them at these points

accordingly.

The baud rate generator module shown in Figure

4 generates an oversampling signal with frequency equal
to 16 times the configured baud rate. This signal is

employed as enable ticks to the UART receiver in order

avoid creating a new clock domain and violating the

synchronous design principle. Assume that the

communication uses N data bits and M stop bits. The

oversampling scheme works as follows:

a. Wait until the receiving of the start bit then start

the sampling counter.

b. After the counter reaches 7, this indicates the

middle point of the start bit. Clear the counter to

0 and restart.

c. When the counter reaches 15, this indicates the

middle point of the first data bit. Retrieve its

value and shift it into receiving register, then

restart the counter.

d. Repeat step 3 N-1 more times to retrieve the

remaining data bits.

e. If the optional parity bit is used, repeat step 3 one

time to obtain the parity bit.

f. Repeat step 3 M more times to obtain the stop

bits.

The receiver block consists of a finite state

machine that has 4 states, at state S0 the receiver waits

until the data pin equals to zero. This indicates the start

of transmission then it goes to state S1. At this state the

receiver counts from 0 to 7 to be sure that the sampling

will be exactly at the middle of the received bit as

discussed earlier, then it goes to state S2 where it stores

the incoming bit in a shift register each time the counter

reach 15. After storing the 8 data bits which is indicated

by the bit counter the state machine goes to state S3 where

it transfers the 8 bit data to the output port and a load

signal is activated to indicate the presence of new data at

the output port. Figure 5 shows the flow of the FSM used

for the receiver sub-unit.

The receiver interface module shown in Figure 6

consists of a FSM and 4 RAMs each of 32 width and 320

depth (256 data + 64 cyclic prefix). As the data received

from UART is arranged in 8 bits a FSM is required for

arranging the incoming data into 32 bit words to be stored

in each location of the RAM. Therefore the finite state

machine waits until a load signal from the receiver is

activated then it stores the incoming 8-bit data at the input

port in a 32bit shift register. This process is repeated 4

times until the 32 bit shift register is full then it asserts a

write enable signal to the corresponding RAM to store the

32 bits data and increments the address for the next

iteration. If the address reaches 320 the FSM switch to the

next ram, until all rams are full after storing all data the

FSM goes to the reading state where it starts reading from

all RAMs at the same time from address 0 to 319. This

gives the MIMO receiving sub-system a stream of data

which is exactly the same as the data arriving from the

two antennas in case of 2×2 MIMO-OFDM system.

5. UART TRANSMITTER

ARCHITECTURE

Similar to that of the receiving subsystem, the

architecture of the UART transmitting subsystem is

shown in Figure 7. It consists of a transmitter, baud rate

generator, and interface circuit. The transmitter is built as

a shift register that shifts out data bits at a rate equal to the

baud rate. The baud rate generator produces one-clock-

cycle enable ticks to control the transmission rate. The

frequency of the ticks is 16 times slower than that of the

UART receiver. Instead of introducing a new counter, the

UART transmitter usually shares the baud rate generator

of the UART receiver and uses an internal counter to keep

track of the number of enable ticks. A bit is shifted out

every 16 enable ticks.
The transmitter consists of a finite state machine of only 2 states, at the initial state if the done signal is received from the interface circuit is activated the transmitter stores the 8-bit data at its input port in a 9 bit shift register and it also sets the LSB in the register to 0 to be the start bit and at the same time the data output port is equal to 1 to indicate that there is no transmission operation yet.

After this, the state machine switch to the next state where it waits until it receives a signal from the baud rate generator which means that it has started to shift out the data in the shift register bit by bit to the data port. A counter indicates how many bits are shifted out from the shift register when it reaches 9 (this means that all data has been shifted out) and returns back to the initial state then reset the data port to 1.

The interface module for the transmitter subsystem takes the output of the receiver which is the binary data symbols and arranges them in a shift register that represents the received data from both antennas. Then, it starts sending the data stored in the register to the UART. This is done using a FSM controller that receives both output streams from the MIMO receiver-subsystem and stores them in the shift register. At the initial state it waits until data ready signal is received from the MIMO receiver subsystem. Then, it starts shifting the received data in a shift register 2 bits by 2 bits in case of 2X2 MIMO-OFDM system and a counter indicates the number of shifts. When it reaches 48, then it starts sending 8-bit packets of data to the UART transmitter with enable signal to enable the transmitter to start sending and waits the done signal from the transmitter that indicates the completion of transmission of the 8-bit data then it send the second 8 bits. This process is repeated until all the data are transmitted.

6. IMPLEMENTATION REUSLUTS & SIMULATION

The complete UART sub system used for testing the implemented design performs the required function of testing the circuit in real time and verifying its operation with small amount of consumed FPGA resources, Table 5.2 shows the required resources for adding the UART sub system to the main design and the amount of consumed resources from the vertex 5 FPGA.

The UART subsystem is tested using VHDL test benches for both transmitter and receiver subsystem. Figure 7 shows the receiver subsystem simulation results. A VHDL test bench sends a hex 55 serially to the input of the receiver using the RS232 protocol with a baud rate of 19200 and the results shows that the receiver successfully extracted the 8 bit signal from the received waveform.

A VHDL test bench is written for testing the UART transmitter subsystem in figure 8 the test bench sets the input signal to hex AA and the simulation results
shows that the 8 bit signal is transmitted successfully using the RS232 protocol.

### Table 1: UART subsystem consumed resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Consumed number</th>
<th>Percentage of Vertex 5 resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>320x32-bit single-port RAM</td>
<td>4</td>
<td>1%</td>
</tr>
<tr>
<td>4-bit adder</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8-bit adder</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9-bit adder</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>12-bit adder</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1-bit register</td>
<td>9</td>
<td>1%</td>
</tr>
<tr>
<td>12-bit register</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>32-bit register</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4-bit register</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1-bit latch</td>
<td>1</td>
<td>1%</td>
</tr>
<tr>
<td>8-bit latch</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Fig 7:** UART receiver subsystem simulation results

**Fig 8:** UART transmitter subsystem simulation results

**REFERENCES**


