OFDM Modem Implementation Using the DSK TMS320C6416T

Miguel Mateus, MSc student

Instituto Superior Técnico. Universidade Técnica de Lisboa, INESC-ID R. Alves Redol 9, 1000-029 Lisboa, Portugal

Abstract—Orthogonal Frequency Division Multiplex (OFDM) modems are everywhere, from wireless networks (802.11a/g/n/ac norms) to Long Term Evolution (LTE), in Power Line Communications (PLC) or Terrestrial Digital Video Broadcasting (DVB-T). Although it is a technology born in the 60’s, it has found greater attention over the past 15 years, due to miniaturization of hardware specific implementations, which led to cheap, widely implemented OFDM modems. Additionally, in these last few years, there has been an effort to use Software Defined Radio (SDR) on more generic CPUs, like the Texas Instruments TMS320C6416T, to provide a flexible way to change the modem behavior simply by changing its firmware.

This work discusses the architecture and implementation options for an OFDM modem in a TMS320C6416T DSK, by looking at some simulations, flowcharts and experimental results. It uncovers some interesting challenges faced and presents the solutions used, such as how the use of a FIFO for the stabilization of the program information flow and how interruption code minimization played a key role in allowing more clock cycles to be available for the main blocks. These results will help everyone who is planning on building any communication system from scratch.

At the end, a modem core capable of transmitting up to 52.2kbps of data was achieved (limited only by the USB interface).

Index Terms—OFDM modem, TMS320C6416T, OFDM implementation, DQPSK, scrambler.

I. INTRODUCTION

The first OFDM theoretical models date back to the sixties, but it was only recently (around the nineties) that they started to find practical applications, since in the past there was not enough processing power [5].

Today, almost every wireless standard, such as wireless LAN 802.11a/g/n/ac, the fourth generation mobile phone LTE/WiMAX standards or the DVB-T/DVB-H/DAB normalizations for Digital Video/Audio Broadband, use OFDM based schemes to transmit and receive data from the channel. Also, some very important wired standards, like the Digital Subscriber Line technologies (ADSL, HDSL or VDSL) or the Power Line Communications applications (PLC) rely on this scheme to transmit information, therefore it is a very popular and important transmission scheme.

A lot of work has already been done in the OFDM area and SDR implementations have been made (a popular one is provided in GNU Radio[1]). Conceptually, references [5] and [11] are recommended, since they provide a strong theoretical insight into OFDM concepts (and provide also a more detailed OFDM history).

There are some known OFDM communication systems implementations in DSPs. An example is the project in reference [1], developed for the TMS320C5510. Other articles like [16] developed for the TMS320C6413 and [8] or [15], focus more on results and conclusions on how a better performance can be achieved.

In this work a baseband OFDM modem was implemented, featuring differential coding, bit scrambling, a QPSK constellation, cyclic prefix handling and a dual sliding window detection algorithm. For further details in any topic on this article, please consult reference [7].

II. OFDM PRINCIPLES

A. Quick technical overview of OFDM

OFDM is closely related to frequency division multiplex (FDM). This old transmitting scheme is based on transmitting several links in parallel to increase the data throughput. In OFDM the concept is the same, but it uses orthogonal subcarriers instead, accomplishing less spectrum usage, thus increasing spectral efficiency.

FDM required a significant guard band to avoid frequency leakage from neighbor subcarriers, thus wasting bandwidth. In OFDM subcarriers are overlapping each other.

This overlapping is possible by adjusting each subcarrier spectrum (which has a sinc(x) = \( \frac{\sin(\pi x)}{\pi x} \) shape) to exhibit its maximum when all its neighbors sinc is null, therefore each subcarrier waveform is not affected by any of its neighbors (see Figure 1). This means each subcarrier is independent (or orthogonal) to each other. In Figure 1 the blue lines represent each individual carrier and the red line represents the sum of all of these components. In the red line it is possible to see the Inter

1Available at: http://gnuradio.org (last accessed August 2014).
Figure 1: Comparison between FDM and OFDM symbol spectrum.

Carrier Interference (ICI) caused in the FDM case. In this case the sum of all other subcarrier components at the frequency of any chosen subcarrier is not zero.

OFDM uses the inverse discrete Fourier transform to create a whole OFDM symbol from a set of symbols created for each subcarrier. This OFDM symbol is then recovered by applying the discrete Fourier transform, which converts it back into the original set of symbols that model each of the subcarriers.

In Figure 2 a simple OFDM scheme is depicted and its main functionality divided into smaller blocks. These will be detailed throughout sections IV and V.

Figure 2: Simple OFDM transmitter and receiver diagram.

B. ISI cyclic prefix and ICI

1) ISI and cyclic prefix: Inter Symbol Interference is a major problem in communication systems. This phenomenon occurs when two time adjacent symbols interfere with each other, resulting in two symbols different from the original ones.

Reflections during a signal propagation may cause multipath, which leads to delay spread — a signal dispersion in the time domain. If the time duration $\tau_{au}$ of this disturbance is significant when compared with the symbol interval, the symbol is be distorted by its multipath copies, thus creating ISI. An OFDM symbol occupies a bigger time interval since it divides its bit rate by $N$ subcarriers (i.e. the OFDM symbol is $N$ times longer than each subcarrier symbol), unlike a single carrier system with the same bit rate. Therefore, ISI caused by multipath is much lower in an OFDM communication system.

Introducing a cyclic prefix (CP) into an OFDM frame is a method to simplify the equalization of the channel and fight ISI in a multipath channel. A cyclic prefix is simply a copy of a certain number of samples from the end of an OFDM frame to the beginning of the OFDM frame. The length of the cyclic prefix ($K$) will have to be greater than the length associated with the delay spread for multipath originated ISI to be eliminated.

2) ICI - Inter Carrier Interference: Inter Carrier Interference (ICI) happens when a neighboring carrier interferes with another carrier. For example, take Figure 1. As previous mentioned, it is possible to see ICI in FDM with non-orthogonal subcarriers, as in the OFDM example, there is no ICI — the sum of a subcarrier neighbour spectrum at that subcarrier frequency is null. Therefore, in OFDM, ICI occurs when subcarriers are no longer orthogonal.

ICI can be caused by the Doppler effect or when the channel has a non-linear phase characteristic, for example. OFDM is very sensitive to frequency shifts, so it easily loses its orthogonality in such cases. For more information, consult reference [6].

C. Advantages and disadvantages

OFDM is great for high data rates, with high spectral efficiency, while having multipath on the channel. Since it is a multi carrier scheme, the data stream is divided into several narrow-band, low-rate subcarriers, which enlarges each symbol in the time domain and therefore it is more resistant to ISI than a bit rate compatible single carrier system. This also allows the subcarriers to be less influenced by frequency-selective fading, minimizing the need for an equalizer.

Another great OFDM feature is that multipath fading is easily minimized, as the use of a cyclic prefix allows different subcarriers to arrive with a certain delay between them and still maintaining the orthogonality property.
This minimizes the need for equalizers and simplifies their implementation.

Unfortunately, OFDM has disadvantages too; it has a high sensitivity to frequency drifts and a high Peak to Average Power Ratio (PAPR) [5].

D. Designing an OFDM communication system

In this subsection it will be shown how to calculate some system parameters that are important in the creation of an OFDM communication system.

1) OFDM mathematical description: OFDM is achieved by applying an IDFT to a set of symbols from a given constellation. Mathematically, this means that complex coefficients that define the constellation points \( \varphi_k \) will be the coefficients used to drive each subcarrier \( e^{j \frac{2\pi kn}{N}} \), thus the OFDM signal \( S_n \) definition is simply:

\[
S_n = \sum_{k=0}^{N-1} \varphi_k e^{j \frac{2\pi kn}{N}} \tag{1}
\]

where \( n = 0, ..., N - 1 \) and \( \varphi_k \in \mathbb{C} \).

2) Spectrum Span: OFDM spectrum span can be approximated by its subcarriers main lobe bandwidth. This means that if the bandwidth of a subcarrier is \( W_n \), then the OFDM unilateral bandwidth \( W_{OFDM} \) must be:

\[
W_{OFDM} = N \times W_n / 2 \tag{2}
\]

3) Cyclic prefix design: As explained before (in subsection II-B1), the CP also needs to be chosen carefully to be effective against multipath.

4) Binary transmission rates: This parameter depends on the number of constellation points (M) of each subcarrier and the interval difference between the start of each frame \( T_{OFDM} \). Then, assuming \( M \in \{ x \in \mathbb{N} : 2^x \} \) and that all constellations have the same number of points, i.e. \( M = M_1 = M_2 = ... = M_n \), the bit rate expression is obtained as:

\[
B_r = \frac{N \times \log_2(M)}{T_{OFDM}} \tag{3}
\]

5) Bit sampling and spectrum occupation: A sampling frequency \( F_s \) needs to be set and, in most cases, it is beneficial to set it to the maximum allowed by the hardware, but it can be set to a lower value if, for example, a higher sampling frequency leads to performance issues. This frequency will define the maximum baseband frequency allowed, as by the low-pass filter Nyquist theorem (shown in Equation [4]):

\[
F_{max} = \frac{F_s}{2} \tag{4}
\]

Also, it can be shown [7] that the subcarrier frequency is given by:

\[
f_{sc} = \begin{cases} \frac{k}{N} F_s & , k < \frac{N}{2} \\ (1 - \frac{k}{N}) F_s & , k \geq \frac{N}{2} \end{cases} \tag{5}
\]

where \( k \) is the index of the subcarrier and \( f_{sc} \) is the frequency of that subcarrier.

6) Pilot subcarriers: OFDM communication systems normally use pilot frequencies, which carry timing and frequency information to help the receiver synchronize with the transmitted signal. There can be also subcarriers to serve as frequency guard (to protect against ICI or for example).

III. System Architecture

This project used a Texas TMS3206416T fixed point processor based DSK, which is a 1GHz device capable of delivering up to 8000 million instructions per second (MIPS) and is designed for applications that require a high performance DSP [10].

Some features of the TMS320c6416T DSK can be found in [12].

Also, this board allows a computer to interact with the DSP to configure it through a built-in USB interface, or to send data to it, using a daughter card [3]. This daughter card has support for parallel port, RS232 serial port and USB emulated serial UART. The board has a stereo audio CODEC TLV320AIC23, which has a DAC and an ADC per channel and they support 96, 48, 44, 32, 24, 16 and 8kHz sampling rate. More information about the system architecture can be found on references [3], [10], [12] and [13].

IV. Transmitter

A. Overall program structure

It is important to keep in mind that this CPU is able to run with a performance of up to 8000 million instructions per second (MIPS) at a clock rate of 1000 MHz [12], but the audio CODEC data input and the USB UART data input have an interrupt update rate of 48kHz and 1kHz, respectively. This means a balance has to be found, so that neither the CPU is blocked waiting for samples, nor the CODEC/USB interruptions are held by CPU during long code executions. The solution is to check for interruptions while the CPU is in such routines and store those interruption values into variables that can be read later. This greatly increases performance.

The implemented main running loop was divided into 4 handlers to better control the program flow (displayed in Figure 3). These are:

- The USB and the CODEC handlers, which are independent and work with a serial dataflow. They attend the interrupt functions, reset their respective flags and exchange data between interrupts variables and their internal queues.
The reception and transmission handlers, which do all the processing activities, concentrating the 'heavier' tasks and calling USB and CODEC handlers to break down long code executions.

The program flow is further explained in subsections IV-B and V-C.

Figure 3: Flowchart illustrating the modem’s main loop.

B. Transmitter flow design and transmission handler

OFDM communication systems need to handle information in serial and parallel structures and converting from one to another requires data buffering up to a certain amount. This buffering needs special attention, because it needs to be controlled such that there are no overflow or underflow problems. The kind of structures and processes that are used to convey the information are depicted in Figures 4 and 11 for the transmitter part and the receiver part, respectively (the receiver flow is addressed in subsection V-C).

Figure 4: Implemented transmitter data flow solution.

The whole transmitter data flow is displayed in Figure 4. When an interrupt is generated, the USB handler will attend the interruption as soon as it is triggered. When this happens, it evaluates if there is space in the 'USB in FIFO' and, if there is, it stores the information in it. This data structure is a 'first in first out' (FIFO) queue for data and it was implemented as a circular buffer FIFO, because it is very important to recycle the memory allocated space in the board.

This circular buffer is an array which recycles the oldest values as new ones arrive (for more information consult [14]).

Once the transmission handler is triggered, the 'USB in FIFO' data elements are consumed to produce DQPSK elements, which have a real and an imaginary part. These are stored in the 'IFFT FIFO'. This FIFO is different from the circular buffer described earlier, although it also recycles the previous addresses. In this case, whole arrays with size $2^N$ are used as the elements in the circular buffer, so there are also reading and writing pointers to data values, but once one array ends, they will point to the beginning of another array. This allows conversion from a serial flow, to the parallel flow needed for the IFFT.

When there are enough values in an array (and there is enough space in 'TX FIFO') the IFFT algorithm runs and it outputs its values to an array. This array is then copied to 'TX FIFO' (which is a buffer of the same type as 'USB in FIFO').

Through experimental results, it was concluded that it is a good idea to have more than 2 arrays in 'IFFT FIFO', because otherwise overflows were observed. As the input speed reaches the maximum output speed, there was just not enough time to process one entire array before the next, thus the handler would stop which caused USB to drop data.

The CODEC handler is always waiting for a CODEC interruption to run. When this condition is met, it then evaluates if it has information available in 'TX FIFO'. If it does, it will copy this information to the interrupt buffer.

Initially it was thought to be possible to skip CODEC and USB handler functions, by having the interrupt instructions to write directly in the circular buffers. This turned out to be very difficult due to the restrict amount of instructions the interruption handling section can hold and this approach was abandoned.

C. Bit scrambler

Scrambling data is very important because it can be used to eliminate long sequences of 0’s or 1’s when these appear in the input sequence (due to the pseudo random nature of information, they do eventually appear), therefore eliminating undesirable data patterns that can cause problems with the receiver [2].

The most basic scrambling function consists in changing the order of the incoming bits with a pre-determinate rule, rendering the message pseudo-random. For more types of scrambling techniques and functions and further information on this subject, please consult reference [11].
A simple way to perform scrambling in a real time application is to use a XOR function and a set of memory storage blocks. The implemented scheme is mathematically described by the polynomial:

\[ P(x) = 1 \oplus x^6 \oplus x^7 \]  

(6)

This scrambler algorithm has a good performance for this project, it is relatively fast and it is easy to implement, therefore it was chosen.

D. Differential coding

Due to the channel physical properties, the transmitted signal may reach the receiver inverted or, more generally, with an arbitrary phase lag. This is why differential coding is important. By using this technique, each constellation symbol is chosen based on the previous symbol.

It does this by coding the phase difference and not the PSK phase itself (see example in Figure 5), without adding any information redundancy (only at the expense of the first few symbols, in case of an inversion).

\[
\begin{align*}
\text{Right: 01, Same: 00} \\
2 \times \text{Right: 00, 3 \times \text{Right: 10}}
\end{align*}
\]

Figure 5: DQPSK construction rule and some examples.

To implement M-DPSK coding, consecutive \( \log_2(M) \) bits were grouped, forming a number. This number (that would index a PSK symbol array if the differential coding block was bypassed) is then processed as the diagram of Figure 6 shows.

E. Constellation mapper implementation

These M-DPSK symbol indexes are used in the M-PSK constellation mapper to index an array containing the pre-calculated values of the real and imaginary parts of the constellation symbol.

Figure 5 shows how a Gray coded DQPSK constellation can be created. Figure 6 details the whole procedure, from bit grouping to complex values conversion.

F. OFDM modulator

The OFDM modulator part has the responsibility to convert constellation symbols into OFDM symbols and then add the cyclic prefix.

OFDM modulation is accomplished using the IDFT properties, which allow for a set of complex numbers to be interpreted as frequency coefficients. After processing the DFT (using an FFT algorithm) these frequency coefficients are converted into time domain samples and an OFDM symbol is created.

The cyclic prefix is used to mitigate ISI caused by the delay spread associated with the channel. It can be easily implemented by copying the last \( K \) samples of the OFDM symbol to the front of the OFDM symbol.

The IFFT algorithm was imported from the DSPLIB\(^2\) libraries for TMS320C64xx because this function (developed by Texas Instruments) was created for this processor and it is already tested and optimized for this kind of hardware. More specifically, function \textit{DSP_ifft32x32} was used (and \textit{DSP_ifft32x32}, for the FFT used at the receiver).

G. OFDM symbol grouping

One OFDM frame can group several OFDM symbol data, which increases bit efficiency (by reducing guard intervals between frames) or it can be divided into smaller ones.

These frames should not be very big, because bigger frames require more data per frame, thus increasing latency and the sizes of the buffers. Still, it requires smaller buffers than those needed if the IFFT/FFT window is increased, so it is still a good choice to increase bit efficiency.

Information that needs to be written to the channel use the Codec handler. It reads elements from ‘TX FIFO’ and delivers them to a buffer that is read during CODEC interruption.

In this work, only one symbol per frame was chosen, because the bit rate bottleneck is elsewhere (see receiver

\(^2\)The source code and compiled libraries are freely available for download at: http://processors.wiki.ti.com/index.php/DSPLIB (visited on August 2014).
results). However, the implementation allows this to be changed easily.

H. Results

1) Specifications for the achieved modem: This sub-section summarizes the conditions for the default configuration.

The results for this work (both transmitter and receiver) were all collected using the ‘Default’ specifications described in the Table I (unless otherwise stated in the result description).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Default</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of subcarriers (N)</td>
<td>64</td>
<td>4,8,16,32,64,128,256*</td>
</tr>
<tr>
<td>Sampling rate</td>
<td>48kHz</td>
<td>96,48,44,32,24,16 and 8 kHz*</td>
</tr>
<tr>
<td>OFDM symbols per frame</td>
<td>1</td>
<td>up to 10*</td>
</tr>
<tr>
<td>Modulation type</td>
<td>DQPSK</td>
<td>DQPSK, QPSK</td>
</tr>
<tr>
<td>Gray code</td>
<td>Activated</td>
<td>Activated/Deactivated</td>
</tr>
<tr>
<td>Scrambling</td>
<td>Activated</td>
<td>Activated/Deactivated</td>
</tr>
<tr>
<td>Preamble</td>
<td>Deactivated</td>
<td>Activated/Deactivated</td>
</tr>
<tr>
<td>Cyclic prefix</td>
<td>4 samples</td>
<td>up to N</td>
</tr>
<tr>
<td>Zero padding</td>
<td>Deactivated</td>
<td>up to N</td>
</tr>
<tr>
<td>Pilot subcarriers</td>
<td>Unused</td>
<td>Unsupported</td>
</tr>
<tr>
<td>CFO estimation</td>
<td>Unused</td>
<td>Unsupported</td>
</tr>
<tr>
<td>Timer 2 (TX)</td>
<td>34/Fs</td>
<td>Higher than timer 2 (RX)</td>
</tr>
<tr>
<td>Timer 2 (RX)</td>
<td>20/Fs</td>
<td>Minimum tested: 20/48kHz</td>
</tr>
<tr>
<td>T_{OFDM}</td>
<td>2.083ms</td>
<td>*</td>
</tr>
</tbody>
</table>

Table I: Configurations used in this work and additional features supported. The entries marked with * depend on other parameters and certain configuration modes may not be possible due to performance restrictions.

2) OFDM spectrum results: In Figure 7 the spectrum of some OFDM frames, randomly internally generated is presented. The spectrum is shown in green and the real and imaginary signals that originated it are also shown, but for reference only. As expected, the spectrum of such set of frames is nearly flat.

3) Single subcarrier spectrum: As described earlier, this OFDM communication system uses DQPSK over orthogonal subcarriers. Considering the example represented in Figure 7 a single subcarrier frame was generated (subcarrier 15). The result is shown in Figure 8.

Figure 8: Result for a single OFDM subcarrier.

According to equation 2 a single OFDM subcarrier in these conditions should have a bandwidth of:

\[ W_n = \frac{W_{OFDM} \times 2}{N} = \frac{48kHz \times 2}{64} = 1.500kHz \]  \hspace{1cm} (7)

When measuring the bandwidth \( W_n \) in Figure 8 a value around 1.5kHz is obtained, confirming the result obtained theoretically. Also, it was expected to obtain a sinc shaped signal for any subcarrier, as seen in Figure 1 of section II. This is also observed in this result.

Taking into account this measurement, then for a ‘pure’ DQPSK scheme applied to N subcarriers:

\[ B_r = \frac{W_n \times N \times \log_2(M)}{2} = \frac{1500 \times 64 \times 2}{2} = 96kbps \]  \hspace{1cm} (8)

would be a valid bit rate expression. However, in the case studied it is not applicable, because OFDM sends information in packets and those frames contain a guard time between them. Also, there is present a time interval associated with CP and an interval associated with preamble could also be present. All these kind of time intervals cause some overhead and that needs to be accounted for as well. The equations in subsection II-D4 reflect these overheads. In this case:

\[ B_r = \frac{N \times \log_2(M)}{T_{OFDM}} = \frac{64 \times 2}{2.083} \approx 61.24kbps \]  \hspace{1cm} (9)

where \( T_{OFDM} = 2.083ms \) can be calculated from Figure 8 by noticing there are 24 frames in 50 ms (50ms/24 = 2.083ms). However, it is demonstrated in 7 that decreasing these overheads it is possible to achieve values close to 90kbps.
4) Scrambler results: In [7] a comparison between the spectrum of a scrambled and an unscrambled frame is presented. From those results it was shown that activating the scrambler leads to lower OFDM PAPR and that frame detection without preamble becomes easier.

V. RECEIVER

Similarly to the last section, in this section the different key blocks of the receiver will be explained and at the end some results will be presented.

A. Frame synchronization

This modem does not transmit continuously. Due to the IFFT/FFT parallel nature, OFDM frames (containing one or more OFDM symbols) are created, thus time guards have to be inserted in between frames.

In Figure 9 the frame synchronization issues are shown. There are essentially 3 important synchronization moments, the one defined by the start of the frame, the end of the frame (defined by timer 1) and the guard time (defined by timer 2).

![Figure 9: How frame detection is performed.](image)

However, the whole synchronization depends on how the frame is detected, since all the counters rely on this event to start. The consequences of a small delay in the detection can corrupt the whole frame.

B. OFDM frame detector

To use the FFT on the received signal, it is necessary to first determine its boundaries. Therefore, a precise, fast and preferably easy to implement algorithm to detect the frame should be used. A solution is to use the sliding window algorithm depicted in Figure 10.

This algorithm consists in two windows, one ahead (A) of the point where the algorithm is analyzing the start of the frame, the other one behind (B). These windows average the square value of the samples. By comparing A and B values, most detection events are relative to the OFDM signal initial peeks and not due to a slow increase in the mean energy value of the channel. This algorithm introduces latency, as when a frame is detected the samples that are analyzed are the delayed samples coming from the buffer associated with window A, not those coming directly from the CODEC interrupt. This algorithm is further explained in [11].

C. Receiver flow design

The flow at the reception is very similar to the flow at the transmitter, described in subsection IV-B where some of the concepts used are already explained. It starts when there is a CODEC interruption, as illustrated in Figure 11.

![Figure 10: Sliding window algorithm examples.](image)

![Figure 11: Implemented receiver data flow solution.](image)
FIFO’. If this is the case and ‘USB out FIFO’ is not full, then it runs the OFDM demodulator, constellation demapping and all the other reception blocks. The result is then stored in ‘USB out FIFO’.

When the USB handler runs, it has a similar behavior to the CODEC handler. It first processes the transmitter part, then it checks ‘USB out FIFO’ for new data and, if any, it outputs it to the ‘USB buffer’.

D. Receiver handler implementation

As mentioned before in section IV, the main loop of the modem program has 4 important handlers which handle everything (Figure 3).

When there are enough symbols in the ‘Rx FIFO’ and there is enough space in ‘FFT FIFO’, the receiver handler runs. It uses the symbols collected to perform a FFT, which recovers the DQPSK information. Until the whole OFDM symbol is not finished, it continuously estimates each DQPSK symbol, performs the necessary decoding operations, converts the symbols into bits and performs the descramble operations. In between some of these operations, USB and CODEC handlers evaluators are called to avoid losing any interrupts from the CODEC or the USB.

E. OFDM demodulator and constellation demapper

An FFT is used to demodulate the OFDM signal, as opposed to the IFFT which is used in the transmission.

At the output of this function, each initial DQPSK symbol (modulated on each subcarrier) will be obtained. However, noise will be present. Also, from the DFT definition, the DQPSK symbols require rescaling to have the same amplitude as the original symbols, so this is also done.

In the demapping process a lookup array cannot be used to convert the symbols received directly, since the channel has introduced noise to the transmission. Therefore the symbol does not have exactly the same value as it once had when it was transmitted. However, the symbol is likely to be found near its transmitted value; as a result, QPSK constellation demapping can be done by placing decision boundaries at the real and imaginary axis of the constellation, as shown in Figure 12.

In this project a DQPSK detector was developed. This is divided into three steps: the QPSK constellation demapper, the Gray code decoder and the differential decoder.

After estimating which symbol number was received, this estimate passes through a Gray lookup array, which was generated at setup time since it is difficult to implement an algorithm faster than this array access (see Figure 12). After this block, the symbols go through the differential decoder and then through a lookup table, which converts symbols into bits.

F. Differential decoding

Differential decoding is the inverse operation of the differential coding explained in IV-D. It allows the signal to be reconverted back from the codification introduced in the transmitter to minimize the impact of a signal inversion. This means the constellation used in each subcarrier will use its previous subcarrier value to determine if the signal is inverted or not. This allows an easy alternative to the use of pilot subcarriers, although it does not replace all their advantages over this technique. To implement M-DPSK decoding, the constellation index is used as suggested in the code presented in Figure 12.

G. Bit descrambler

The bit descrambler is the reverse mechanism to the bit scrambler. It recovers the original unscrambled sequence by passing the bits recovered from bit ungrouper (illustrated in Figure 12) through the function defined by the polynomial in Equation 10

\[ P(x) = 1 \oplus x^5 \oplus x^6, \] (10)

which is then delayed one more time to form the correct unscrambled sequence. Then, as illustrated in Figure 12, bit sequences are grouped into char sequences and these are stored in ‘USB out FIFO’, which is read by the USB handler.

H. Receiver Results

1) Results for the reception capabilities: The whole system was tested using a program written especially to test the speed of a loop back connection in the USB emulated serial port. This can be found at [7].

To characterize the efficiency of the main handlers in transmitting information, three situations, illustrated in Figure 13 were tested:

1) The maximum bit rate allowed by the serial UART functions used in the DSP.
2) The maximum bit rate without errors when an internal loop is performed.
Figure 13: Figure illustrating the three loops used in their respective three situations.

3) The maximum bit rate without errors when an external loop is performed or, in case of errors, the percentage of errors at the receiver when the bit rate is near the bit rate of the internal loop.

The first test is to determine a baseline for comparison. For this it was decided to measure the maximum bit rate allowed by the USB emulated serial UART functions used in the DSP, when the main loop is stripped down from all handlers and these are the only functions called. By doing so, it was determined that it is possible to achieve up to 64kbps, which is about 64/737.280 = 8.68% of UART’s transmission capabilities.

The second test was to evaluate the program maximum bit rate when transmitting and receiving without errors. This was done first by performing an ‘internal loop’, which means copying all the samples inside the CODEC handler, from its output to its input, with the objective of bypassing the DAC, the channel and the ADC. After achieving these modifications, a bit rate of approximately 52.2kbps was measured. This result is about 52.2/64 = 81.6% of the maximum allowed by the DSP, as measured by the first experiment.

The last experiment done in the reception was to do the same as in the second test using an external loop (a cable connected from the CODEC’s input to its output), but under similar conditions, it was observed about 60% errors in 8 bit words (and about 25% BER). To understand why this was happening, some of the intermediate values related to an OFDM frame (1 frame analyzed per each run of the program) were stored in its internal memory. The values stored allowed to re-create the original transmitted and received OFDM frames and the DQPSK constellations at the transmitter and receiver. One example is shown in Figure 14.

In Figure 14a the received signal’s QPSK constellation is shown. In here it is seen that the constellation is rotating when the loop is done using an external cable (Figure 14b), as for when the loop is done internally there is no such issue (Figure 14a). Also, some symbols are closer to the center of the constellation (due to non-linearities in the CODEC).

The rotation of the signal is due to the different clocks operating the AD and DA of the CODEC. Because the receiver does not have any synchronization algorithm (frequency offset, phase delay and symbol timing), this results in a very large bit error rate.

VI. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this work, the design, development and implementation of an OFDM modem was conducted for a fixed point TMS320C6416T DSP. This modem was tested with 64 DQPSK data subcarriers, with Gray coding and scrambling activated, where each OFDM frame carried only one OFDM symbol with a 34 sample separation between frames. Under these conditions, the maximum theoretical value for the bit rate of the OFDM system is 61.24kbps.

However, the bit rate of USB UART to and from computer was tested by programming the DSK to solely copy the values received at the USB interruption to its USB output, making a loop. The maximum bit rate measured under these conditions was 64kbps, far from the 61.24kbps theoretical limit (and about 8.68% of its 921600 baud rate configuration). As a consequence of the data flow control limitations deriving from this result, the main conclusion for this work is that the modem was capable of achieving 52.2kbps reception rate which is, however, almost 82% of the maximum USB UART bit rate. Other conclusions were taken: these were grouped into the two subsections that follow.

1) Design and information flow: A first conclusion is that for each transition from serial to parallel, or vice versa, a buffer needs to be added. This conversion needs to be as efficient as possible, as these operations are done often. Then, most of the processing work has to be done in parallel to the interruption handling and so it is important to pay attention to the flow control. The handlers should be able to store the information regardless if another value in the queue is going to be used by another handler and this memory space should be recyclable, since is constantly being used.
FIFO queues have to be designed to avoid overflow of information. They play a very important role linking the logical blocks, since they control the information flow in the program. Additionally, in this DSK the USB handler takes some time to run. To ensure CODEC samples will not be missed, when the USB handler is called it calls the CODEC handler.

Finally, it was concluded that the CODEC and USB handlers should not be included inside the interruption area, because of the limited space available there. This current design also presents the advantage of easy down-scaling to slower and cheaper DSPs.

2) Transmission and reception: Several conclusions were taken while studying and performing tests on the transmitter and receiver. For example, while studying the OFDM subcarrier bandwidth occupancy, it was found that the theoretical scheme used only allows 61.24kbps, as opposed to the rate of 64 DQPSK carriers with the same spectrum characteristics, which support about 96kbps. The difference is an efficiency issue, caused because OFDM uses frames, so there are guard times and preambles. However, this can be easily overcome by increasing the number of OFDM symbols per frame or increasing the number of subcarriers. Although these features were implemented, there was no need to do this because the bottleneck was found to be in the USB side, as already mentioned.

Finally, after connecting the input to the output with a cable and activating these blocks (loop 3), errors were found. While studying the constellation received, it is observed that there are some samples with smaller amplitudes (which are due to the typical non-linearities found in audio DAC/ADC input/output filters) and that the DQPSK constellation rotates (which is due to the different clocks operating the AD and DA of the CODEC). Because the receiver does not have any synchronization algorithm (frequency offset, phase delay and symbol timing), this results in a very large bit error rate.

B. Future work

Some additions to this modem were studied but were not implemented or not fully finished. These are suggested as future work:

- Carrier Frequency Offset (CFO) estimation, which tackles the problem of sample synchronization between transmitters and receivers with independent clock cycles.
- Introduction of pilot subcarriers, which help estimating CFO and simplify equalization, for example.
- A better frame detection algorithm, based on correlation of a known preamble.
- Frequency domain equalization.
- M-QAM and M-PSK implementation, to increase the number of symbols in the constellation.
- Symbol synchronization.
- A redundancy code, to provide error detection and/or error correction.

Although this modem should be used in conjunction with a frequency multiplexer and a power amplifier (with all necessary required adaptation circuits) to be able to effectively serve a more practical application, once CFO estimation is implemented, it should be possible to convert to and from signals in the 24kHz band.

As for hardware proposals, it would be interesting to implement an efficient version of this SDR OFDM modem in a cheaper platform.

Finally, another suggestion would be to create an interface to connect to the TMS320C6416T DSK board to frequency multiplex the signal.

**REFERENCES**