Hardware implementation of OFDM modulator and demodulator using TMS320C6711 DSK board

TOMÁŠ FRÝZA and STANISLAV HANUS
Institute of Radio Electronics
Brno University of Technology
Purkyňova 118, 612 00 Brno
CZECH REPUBLIC

Abstract: - This contribution deals with an OFDM modulator and demodulator implementation in a DSK board. A brief description of the DSK board and Code Composer Studio is presented with the source codes in C language as well. The program of coder and decoder contains several functions, which represent concrete coder's blocs. In this paper, each function is classified by its complexity. On the encoder side, the most time-consuming function is a generation of an input binary frame. This bloc represents almost 80 % computing time of all OFDM encoder functions. On the other side, this function only simulates the input binary stream, thus in the real systems the input frame generation is redundant. On the receiver side, a QAM demodulator has the highest complexity. The number of cycles depends on a FFT length and a type of QAM. For FFT with length of 256 and 64-point QAM constellation diagram, the QAM demodulator consumes 95 % time of all OFDM demodulator functions.

Key-Words: - OFDM, TMS320C6711 DSK, digital communications, FFT, QAM, computing time, Code Composer Studio, C language

1 Introduction

The orthogonal frequency division multiplexing (OFDM) appears in several standards relating to wireless communications at high bit rate, such as terrestrial digital video broadcasting (DVB-T) in Europe or WLAN. OFDM is a special case of multi-carrier modulation, which is the principle of transmitting data by dividing the data stream into several parallel bit streams and modulating each of them onto individual carriers. Each carrier in OFDM system is a sinusoid with a frequency that is an integer multiple of a fundamental frequency. Therefore, the carriers are mutually orthogonal. Sidebands of the individual carriers can overlap but no carrier interference is presented in the received signals [1]. Each carrier is like a Fourier series component of the composite system. An OFDM signal is created in the frequency domain and then transformed into the time domain via the discrete Fourier transform. OFDM modulation can be done efficiently with DSP approach with used of the Fast Fourier Transform (FFT) in the transmitter and receiver.

Section 2 of this contribution is specifically devoted to the description of the basic parameters and equipments of the TMS320C6711 DSK board and Code Composer Studio. The OFDM implementation and source code examples with algorithm's velocity results are given in Section 3. Section 4 discusses achieved conclusions.

2 TMS320C6711 DSK board

The OFDM modulator and demodulator have been implemented in the TMS320C6711 DSP Starter Kit (DSK). This board is shown in Figure 1 and contains a floating point digital signal processor C6711 operates at 150 MHz and delivers an 1200 MIPS (million instructions per second) and 600 MFLOPs (million floating point operations per second). Thus, this tool is powerful and cost-effective at the same time. For additional program and data storage, two external memory are situated on the board: 16 MB SDRAM and 128 KB Flash. Input and output signals are accessible for the DSP through a TLC320AD535 16-bit data convertor.

Research described in the paper was financially supported by the Czech Grant Agency under grants No. 102/03/H109 and No. 102/04/2080, by grant project of the Czech Republic Ministry of Education, Youth and Sports No. 1621/2004 and by the research program MSM 262200011.

Other upgrade can be done by a daughter card interface. The daughter cards are produced for all domain of digital signal processing: from audio processing and finger-print verification to the video compression applications. Easy emulation and debugging is provided by an JTAG controller. The DSK is connected to a PC through a parallel port cable and all communication between DSP and PC is performed by Code Composer Studio.

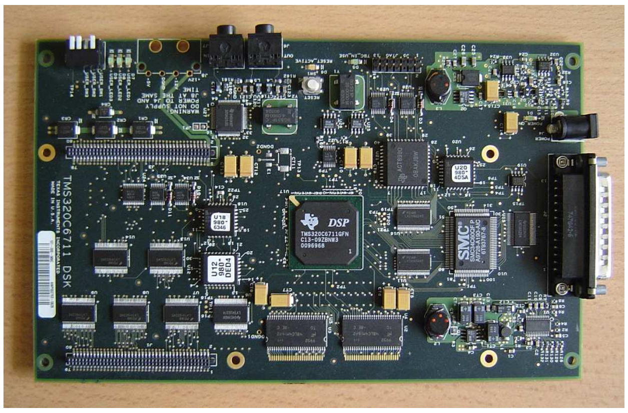

Figure 1: TMS320C6711 DSK board for OFDM modulator and demodulator implementation.

```
int main(int argc, char* argv[]) {
1
2   initOFDM() ;
3   initQAM() ;
4
5   modulate() ;
6
7   demodulate() ;
8   BER() ;
9
10  return 0 ;
}
```

Figure 2: OFDM system main function.

2.1 Code Composer Studio

Code Composer Studio is a development tool from Texas Instruments. This tool includes editor, C/C++ compiler, linker, assembler, simulators, XDS560 and XDS510 emulation drivers, real-time analyzer and DSP/BIOS support. Code Composer Studio allows DSP designers to program, debug, load, control and test any application specified for all digital signal processors from TI.

3 OFDM implementation

This section deals with the OFDM implementation. A source code of the basic functions (written in C language) is presented. In the next text, number in parentheses (1) corresponds with a line in the source code presented in figures.

The fundamental conception of the OFDM system simulation is represented in Figure 2. Note that, there are three basic sections. First is an initialization of OFDM (i.e. a preparation of necessary variables) and Quadrature Amplitude Modulation (QAM) constellation (see lines 2 and 3 in Figure 2). Second is the OFDM modulation (line 5), including generation of input binary frame, next the OFDM coding itself and addition of AWGN channel as well. Last part is the OFDM demodulation (line 7) and evaluation of the bit error rate (BER) performance (line 8). By reason of BER evaluation, lines 5-8 can be plunge in a loop.

```
void modulate() {
1
2   frame_gen() ;
3   QAMcoder () ;
4   iFFT() ;
5   addGuard() ;
6   transmission() ;
}

void demodulate() {
1
2   remGuard () ;
3   fFFT () ;
4   QAMdecoder () ;
}
```

Figure 3: Block scheme of OFDM modulator and demodulator.

A block arrangement of the OFDM modulator and demodulator is shown in Figure 3. The OFDM coder contains succession of the following functions: input frame generation (2), QAM coder (3), inverse FFT (4), addition of a guard interval (a cyclic continuation of the useful part of the symbol, inserted before it) (5) and finally, transmission through the AWGN channel (6). At the receiver side, the OFDM demodulator performs the guard interval removing function (2), forward FFT operation (3) and QAM decoding (4). The input stream generation, inverse FFT routine and QAM decoder functions are closely depicted in Subsection 3.1.

Our implementation allows FFT length of 16, 64 and 256 complex points. The QAM constellation and the length of the guard interval were adopted from the European standard for the DVB-T [2]. Thus, all data in one OFDM frame are modulated using either QPSK, 16-QAM, 64-QAM, non-uniform 16-QAM or non-uniform 64-QAM and the guard interval is defined as 1/4, 1/8, 1/16 or 1/32 of the FFT length. In this paper, the pilot signals are not considered.

3.1 OFDM modulator and demodulator source code

In this section, source code of the OFDM communication system is presented. First, the constants and global variables are outlined, then the most sophisticated functions are described. The constant's and the global variable's declaration is shown in Figure 4, where $nQAM$ is the number of bits used for coding all points in the QAM constellation diagram (i.e. for our configuration, $nQAM = 2$, 4 or 6), α is a constellation parameter ($\alpha = 1, 2, 4$), $nFFT$ is the length of the discrete Fourier transform ($nFFT = 16, 64, 256$), K is the number of used carriers, where $K \leq nFFT$ and $guard$ represents length of the guard interval ($guard = \frac{nFFT}{4}, \frac{nFFT}{8}, \frac{nFFT}{16}$ or $\frac{nFFT}{32}$). A total number of proposed OFDM communication system without consideration of the parameter K is therefore $3 \times 3 \times 3 \times 4 = 108$ settings. In our implementation, the number of unused carriers is equal to zero, thus K represents directly the length of FFT.

The proposed program requires eight global variables (see Figure 4). An array `QAM[2*64]` represents complex values of the QAM constellation. For QPSK or 16-QAM modulation only the first 8 or 32 elements are used. Variables for the FFT calculation `w[3*nFFT/2]`, `x[2*nFFT]` and `y[2*nFFT]` contain a twiddle factor array and input and output complex points arrays, respectively. The arrays `i_coder[]`, `o_coder[]`, `i_decoder[]` and `o_decoder[]` represent input and output frames of the OFDM coder and decoder.

It is known that **`float`** type is represented by 32 and **`char`** by 8 bits. Hence, the total amount of allocated bits in our application can be derived from the following equation

$$\begin{aligned} N \;=\; 4{,}096 \;&+\; 176 \cdot nFFT \\ &+\; (8 \cdot K + 8 \cdot nFFT) \cdot nQAM \qquad (1) \\ &+\; 128 \cdot (nFFT + guard). \end{aligned}$$

Suppose the maximal values of all configure parameters, i.e. $nQAM = 6$, $nFFT = K = 256$ and $guard = \frac{nFFT}{4} = 64$ then quantity of the allocated memory for all global variables is $N = 14{,}336$ Bytes and $N = 55{,}808$ Bytes if $nFFT = 1{,}024$. (Remark: amount of allocated memory do not depends on the parameter α).

```
#define nQAM    4  // nQAM  = 2, 4, 6
#define alpha   1  // alpha = 1, 2, 4
#define nFFT  256  // nFFT  = 16, 64, 256
#define K     256  // K    <= nFFT
#define guard nFFT/4
            // guard = nFFT *
            //(1/4, 1/8, 1/16, 1/32)

float QAM[2*64] ;
float w[3*nFFT/2] ;
float x[2*nFFT] ;
float y[2*nFFT] ;
char  i_coder[K*nQAM] ;
float o_coder[2*nFFT+2*guard] ;
float i_decoder[2*nFFT+2*guard] ;
char  o_decoder[nFFT*nQAM] ;
```

Figure 4: Constants and global variables declaration for OFDM system.

```
void frame_gen() {
1   int i ;
2   float f ;
3
4   for ( i = 0; i < K*nQAM; i++ ) {
5     f = ((float)rand()/RAND_MAX) > 0.5 ;
6     i_coder[i] = (char)f ;
7   }
}
```

Figure 5: Generation of input bit stream.

In the next text, source codes of the most important functions are presented. The input bit stream generation function is denoted in Figure 5. Its object is to load the variable `i_coder[]` by binary data. The `rand()` function from the library `stdlib.h` provides pseudo random integers generation in the range of $\langle 0; \mathrm{RAND\_MAX}\rangle$. Thus, the local variable `f` is equal to 0.0 or 1.0. As we will discover later, this algorithm is one of the most time-consuming functions of the OFDM system.

```
void iFFT() {
1   int j ;
2
3   for ( j = 0; j < nFFT; j++ )
4     x[2*j+1] = -x[2*j+1] ;
5
6   DSPF_sp_cfftr4_dif ( x, w, nFFT ) ;
7   for ( j = 0; j < nFFT; j++ ) {
8     x[2*j] = x[2*j] / nFFT ;
9     x[2*j+1] = x[2*j+1] / -nFFT ;
10  }
11  dig_rev ( x, y, nFFT ) ;
}
```

Figure 6: Inverse FFT function.

Main operation of OFDM modulation is the inverse FFT calculation. This function transfers complex QAM symbols to output OFDM symbols. The OFDM coder and decoder are realized with the aid of the `DSPF_sp_cfftr4_dif` function from the TMS320C67x DSP Library [3]. This library is available on Texas Instruments web page. The routine implements the radix 4 FFT floating-point computation, based on the decimation in frequency. Hence, the number of input points must be a power of 4. This function is an in-place routine in the sense that the output is written over the input, whereas the output is in digit-reversed order. Each real and imaginary input value is interleaved in input array and complex numbers are in normal order `xr[0]`, `xi[0]`, `xr[1]`, `xi[1]`, ... The forward FFT can be done directly with the function `DSPF_sp_cfftr4_dif` followed by output symbols conversion to normal order. The routine can be also used to implement the inverse FFT by performing the complex conjugate on the input complex numbers (see line `4` in Figure 6), and dividing the results by the length of FFT (imaginary values are negated as well `8-9`). Function `dig_rev()` (`11`) provides conversion of digit-reversed symbols [3].

The forward FFT operation in the OFDM receiver is followed by QAM demodulator, which transforms complex symbols to output binary stream. Code source of the proposed algorithm is shown in Figure 7. The algorithm can be divided into three consecutive parts. Purpose of the first one is to reduce the scanned constellation diagram area to the one quadrant only (`10-21`). Second part is a calculation of the distance between received and original symbol (`24-28`). The symbol with the minimal distance is selected as the symbol being transmitted. Last part of the QAM demodulator function is a replacement of the demodulated symbols to the output array `o_decoder[]` (see lines `32-34` in Figure 7).

3.2 Algorithm velocity

For algorithm velocity test, all functions have been analysed by Code Composer Studio tools. The test criterion was number of cycles necessary for execution of all functions. Computation time of each OFDM blocs depends on the FFT length and the type of QAM modulation. Hence, several OFDM settings have been effected. Results (in thousands of cycles) for $nFFT = 2, 16, 256$ and $nQAM = 2, 4, 6$ are shown in Table 1.

4 Conclusion

This contribution deals with an implementation of OFDM modulator and demodulator on a DSK board. Basic description and possibilities of the TMS320C6711 Starter Kit board had been indicated. Code Composer Studio had been presented as well. The main purpose was accentuated to a listing of the OFDM algorithm's source codes. The aim was to present principle of the OFDM modulator and demodulator, implemented in a digital signal processor. The OFDM modulator contains three fundamental segments: a QAM modulator, inverse FFT calculation and a guard interval addition. The OFDM demodulator performs similar operations in inverse order: removing the guard interval, forward FFT computation and finally, QAM demodulator. Source code of several functions had been presented in this paper. All functions had been evaluated by the computation time in numbers of cycles. From Table 1 follows that the most time-consuming functions are generation of input binary data (but in the

Table 1: Computation time of each OFDM coder and decoder function in thousands of cycles versus the type of QAM modulation and length of FFT. The guard interval length was $guard = \frac{nFFT}{4}$ for all test cases.

Computation time [ths of cycles]				
nFFT	16		256	
nQAM	2	4	4	6
initialization	91	142	1,207	1,413
frame generate	107	213	3,408	5,112
QAM coder	7	8	113	233
inverse FFT	28	28	567	567
add guard interval	6	6	74	74
modulate total	148	255	4,162	5,986
transmission	4	4	65	65
remove guard interval	3	3	52	52
forward FFT	23	23	488	488
QAM decoder	49	232	3,767	14,549
BER	4	4	63	63
demodulate total	83	264	4,435	15,217

real systems, this bloc is not used) and the QAM demodulator. These functions absorb almost 80 % and 95 % of modulator and demodulator computing time, respectively. In a future work, especially the QAM demodulator algorithm can be optimized. The modulated signals can be brought out from the DSK board by DSP/BIOS approach as well [4].

Reference:

[1] J. Heiskala, J. Terry, *OFDM Wireless LANs: A Theoretical and Practice Guide*. USA: Sams Publishing, 2001. 314 pages. ISBN 06-723-2157-2.

[2] *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television*. ETSI EN 300 744 v1.4.1. European Standard (Telecommunications series). 2001.

[3] *TMS320C67x DSP Library Programmer's Reference Guide*. Texas Instruments, SPRU657. February 2003.

[4] S. Dirksen, *An Audio Example Using DSP/BIOS*. Application Report SPRA598. Texas Instruments. November 1999.

```
void QAMdecoder() {
  int i, j ;
  char min, max ;
  char ns ;
  float distmin = 10.0 ;
  float dist ;
  char pos ;

  ns = 2<<(nQAM-1) ;
  for ( j = 0; j < nFFT; j++ ) {
    if ( y[2*j] >= 0 ) {
      max = ns >> 1 ;
      min = 0 ;
    }
    else {
      max = ns ;
      min = ns >> 1 ;
    }
    if ( y[2*j+1] >= 0 )
      max -= ns >> 2 ;
    else
      min += ns >> 2 ;

    distmin = 10.0 ;
    for ( i = min; i < max; i++ ) {
      dist = sqrt (
        (QAM[2*i]-y[2*j]) *
        (QAM[2*i]-y[2*j]) +
        (QAM[2*i+1]-y[2*j+1]) *
        (QAM[2*i+1]-y[2*j+1]) ) ;
      if ( dist < distmin ) {
        distmin = dist ;
        pos = i ;
      }
    }

    for ( i = nQAM; i > 0; i-- ) {
      o_decoder[j*nQAM+i-1] = (pos & 1);
      pos = pos >> 1 ;
    }
  }
}
```

Figure 7: QAM demodulator source code.