ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 10, December 2013

Real Time Implementation of OFDM system on DSP Processor

Dr. Seema Verma[1], Pawan Sharma[2]

Associate Professor, Department of Electronics and Communication Engineering, Banasthali University, Banasthali Rajasthan, India[1]

Research Scholar, [2]Department of Electronics and Communication, Banasthali University, Banasthali, Rajasthan, India[2]

ABSTRACT: Orthogonal Frequency Division Multiplexing (OFDM) technology has been adopted by a variety of standards. This technology was chosen due to its robustness at high data rates in a frequency selective multipath channel. This paper describes a real-time implementation of a half-duplex OFDM system using Matlab software simulink, DSP processor TMS320C6713 and Code Composer Studio. In this approach, MATLAB and Simulink make the transition from theory to application easy and enjoyable. The hardware interface converts the baseband signal from the PC to the DSP Processor. Basic transmission and reception performances are evaluated in real time using Real Time data Transfer (RTDX).

Keywords: MATLAB, Simulink, digital signal processing, real-time digital signal processing, RTDX.

I. Introduction

Wideband wireless OFDM communication systems have gained in popularity due to the spectral efficiency and capability of OFDM to transmit high data rates over broadband radio channels with frequency selective fading. Because of its benefit, OFDM technology has been adopted by a variety of standards which include the European Telecommunications Standards Institute (ETSI) for Terrestrial TV, the IEEE 802.11a/g standard for wireless LANs operation at bit rates up to 54Mb/s and the IEEE 802.16a/d for WMAN at bit rates up to 70 Mb/s. With the rapid increase in transistor density, it has become feasible to keep the functionality entirely in a programmable digital signal processor (DSP), allowing much faster changes and upgrades [1]. However, recent DSP technologies have not yet satisfied the requirements of high-speed communication standards. In particular, orthogonal frequency division multiplexing (OFDM) necessary to achieve high-speed data transmission in narrow bands, need to perform several hundred or thousand points of fast Fourier transform (FFT) within a few tens of microseconds. Commercial DSP chips have not yet reached these requirements [2], [3]. High-speed FFT/IFFT computations may be one of the main research topics for the next generation wire/wireless communications. Initially TMS320C5402 DSP starter Kit (DSK5402) is used in [4] and [5] as the combination of C and Assembly programming languages is presented. Latter the main programming language is Assembly. Since Assembly programming might become very tedious, so ready-to-use Assembly code segments are required which they only need to complete and modify as they progress in the implementation. A entirely different approach for creating DSP projects is presented by Gan et al. [6, 7]. Instead of concentrating on C (or Assembly) programming, the students are working with Matlab and Simulink. The powerful Real-time workshop and Embedded Target for TI C6000 DSP toolboxes [8] are used to translate a DSP system to real time hardware. However, in the above references the idea is only shortly exemplified and no attempt is made to create a lab which is entirely based on the Simulink-DSP hardware connection. We describe a technique based on Simulink and Texas Instruments DSK C6713.In this technique we are using Simulink in conjunction with DSP hardware and further expands and enhances it. First, the use of Simulink enables the creation of sophisticated algorithms in an intuitive top-level design. Simultaneously, this approach gives the opportunity to conduct hands-on experiments with real signals and hardware. We tried to focus the efforts on the DSP problems themselves rather than on the actual programming.

II. Related work

An implementation of a half-duplex OFDM system has been proposed by Michael Loughlin et.al[9] using Digital Signal Processing (DSP) LAB 2000 kits by Texas Instruments. Multiple-input multiple-output (MIMO) orthogonal frequency division multiplexing (OFDM) system based on the multi-core Texas Instrument (TI) C64x+ digital signal processor (DSP) has been proposed by Chien Van Trinh et.al[10]. The system is implemented by employing real time

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 10, December 2013

data exchange (RTDX) and serial rapid input/output (SRIO) techniques for communication interfaces between personal computer (PC) and DSP. The simulation and implementation of a complete system on a DSP processor through a graphical programming language, which is SIMULINK proposed by Gihan Gomah Hamza et.al.[11] . This makes every part in the receiver architecture very clear and easier to understand, follow, modify and debug

III. ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING

An OFDM signal can be generated as an N-point inverse discrete Fourier transform (IDFT). In practice, the IDFT can be implemented with the computationally efficient inverse fast Fourier transform (IFFT) as shown in Fig. 1.1. Let $\{S_k, k = 1,2,3, \dots\dots N\}$ represent a block of N complex data symbols the IDFT of the data block is yielding the time-domain sequence $\{ S_n, n = 1,2,3,4, \dots\dots, N\}$.

$$S_n = \sum_{k=0}^{N-1} S_k e^{\frac{j2\pi nk}{N}} , \qquad n = 0,1,2,3, \dots\dots, N-1 , \qquad (1.1)$$

To mitigate the effects of ISI caused by channel delay spread, each block of N IFFT coefficients is typically preceded by a cyclic prefix (CP) or a guard interval consisting of N_g samples, the cyclic prefix is simply a repetition of the last N_g IFFT coefficients. Alternatively, a cyclic suffix can be appended to the end of a block of N IFFT coefficients that is a repetition of the first N_g IFFT coefficients. The guard interval of length N_g is an overhead that results in a power and bandwidth penalty, since it consists of redundant symbols.

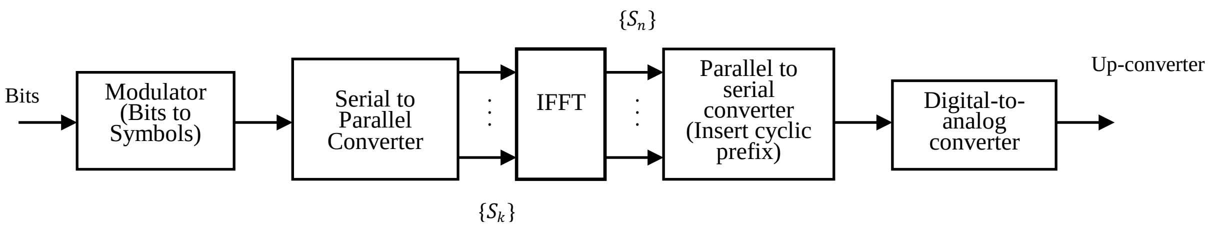

$\{S_n\}$
Bits
Modulator (Bits to Symbols)
Serial to Parallel Converter
IFFT
Parallel to serial converter (Insert cyclic prefix)
Digital-to-analog converter
Up-converter
$\{S_k\}$

Figure 1.1 OFDM using IFFT

However, the guard interval is useful for implementing time and frequency synchronization functions in the receiver, since the guard interval contains repeated symbols at a known sample spacing. The time duration of an OFDM symbol is $(N + N_g)$ times larger than the modulated symbol in a single-carrier system.

At the receiver, the received complex baseband signal is sampled with an ADC, usually with a sampling interval, $\Delta_T = \frac{T_s}{N}$. Sometimes fractional sampling is used, where the sample period is $\frac{1}{M}\,\Delta_T$, where M is an integer greater than one. For simplicity, assume here that M = 1. Then the combination of the DAC in the transmitter, the waveform channel, and the ADC in the receiver creates an overall discrete-time channel with tap spacing AT. After ADC, the N_g samples received during the guard interval of each OFDM symbol are discarded in the case of a cyclic prefix; in case of a cyclic suffix the N_g received samples at the beginning of an OFDM symbol are replaced with the μ received samples at the end of the OFDM symbol. Under the condition that $N_g \geq N_h$, the *linear* convolution of the transmitted sequence of IFFT coefficients with the discrete-time channel is converted into a *circular* convolution. As a result, the effects of the ISI are completely and easily removed. After removal of the guard interval, each block of N received samples is converted back to the frequency domain using an FFT as shown in Fig. 1.2. The FFT operation performs baseband demodulation.

Frequency domain equalization (FDE) is then applied to the N FFT coefficients. Similar to OFDM, the FDE simply multiplies each FFT coefficient by a complex scalar to perform zero-forcing or minimum mean square error equalization. Afterwards, the equalized samples are converted back to the time-domain using an N-point IFFT and applied to a decision device or metric computer. The overall system complexity of SC-FDE is comparable to OFDM. The main difference is that OFDM uses an IFFT in the transmitter and an FFT in the receiver, while SC-FDE does not perform any transformation in the transmitter but employs an FFT/IFFT pair in the receiver.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 10, December 2013

IV. DSP STARTER KIT (DSK) WITH TMS320C6713 FLOATING – POINT PROCESSOR

Digital signal processors such as the TMS320C6x (C6x) family of processors are like fast special-purpose microprocessors with a specialized type of architecture and an instruction set appropriate for signal processing. The C6x notation is used to designate a member of Texas Instruments' (TI) TMS320C6000 family of digital signal processors. The architecture of the C6x digital signal processor is very well suited for numerically intensive calculations. Based on a very-long-instruction-word (VLIW) architecture, the C6x is considered to be TI's most powerful processor. The TMS320C6713 (C6713) is very well suited for numerically intensive algorithms. The internal program memory is structured so that a total of eight instructions can be fetched every cycle. For example, with a clock rate of 225MHz, the C6713 is capable of fetching eight 32-bit instructions every 1/(225 MHz) or 4.44 ns. The C67xx (such as the C6701, C6711, and C6713) belong to the family of the C6x floating-point processors, whereas the C62xx and C64xx belong to the family of the C6x fixed-point processors. The C6713 is capable of both fixed- and floating point processing.

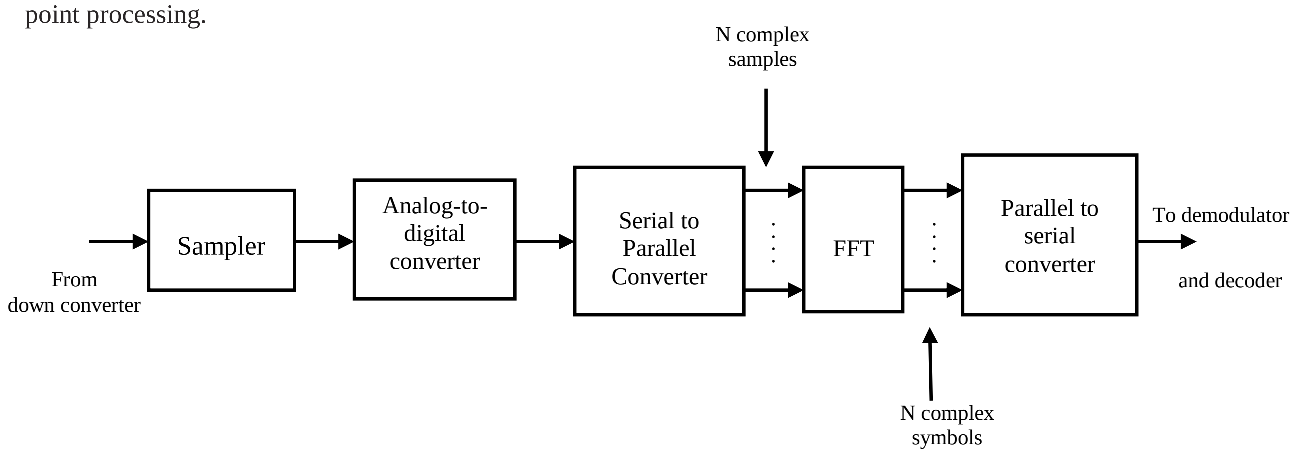


Figure 1.2 OFDM using FFT

A. SYSTEM SPECIFICATIONS

The OFDM system implemented uses a 16- point Quadrature Amplitude Modulation (16-QAM) using look-up tables. Mapping four data bits onto one complex-valued symbol along with Gray coding was implemented making adjacent symbols differ in only one bit. Incoming data bit stream is mapped to a complex number representing subcarriers amplitude and phase. Furthermore, a 64-point Fast Fourier Transform (FFT) and Inverse FFT was implemented.
DSP Board Details:

- TMS320C6713 DSP - 225 MHz, floating point, 256 Kb internal RAM/Cache.
- CPLD - Programmable "glue" logic.
- External SDRAM – 16 Megabytes, 32-bit Interface.
- External Flash - 512Kbytes, 8-bit interface (256Kb usable).

B. DSP IMPLEMENTATION

The block diagram in figure 1.3 features the interface of the Simulink model developed with MATLAB and CCS (Code Compose Studio) IDE along with the Target specified in the model. Here the Target we have used is the TMS320C6713 DSP kit as seen in the figure. RTW and Embedded Target Coder tools on Simulink generate ANSI C code which is automatically dumped onto the CCS IDE.

V. DSP SOFTWARES

The TI C6713 DSK, like a personal computer, also needs to load software to establish its behaviour and function. These software's can be designed in a variety of approaches. The design approach that the real time implementation will follow is illustrated in Figure 1.4. The designer will begin with a concept of what they want to program. The next step is to model the concept with blocks from Simulink's large collection of prewritten blocksets. Basically, a block diagram that models the concept is built using Simulink. If a specific block required is not included in Simulink's blocksets, the designer may choose to write their own blocks from scratch using Matlab. At this point the design is still not designed to operate on any specifically hardware. To do that, the designer uses the C6x Target and RTW to generate (or build) ANSI C code intended for the TI C6713 DSK. The C6x Target will then automatically take the generated ANSI C code and uses the TI CCS tools to compile specific machine code and finally loads the targeted machine code to the TI C6713 DSK hardware. For the experienced designer, they may choose to directly write the ANSI C code.

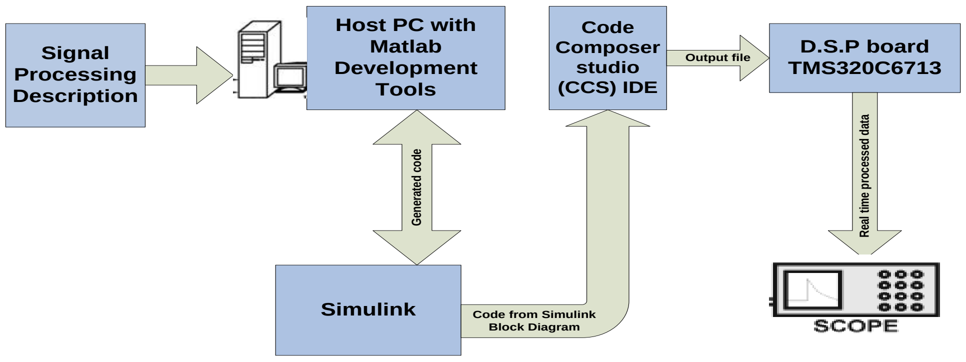

Signal Processing Description
Host PC with Matlab Development Tools
Code Composer studio (CCS) IDE
Output file
D.S.P board TMS320C6713
Generated code
Real time processed data
Simulink
Code from Simulink Block Diagram
SCOPE

Figure 1.3: Block diagram of Interface Simulink model developed with MATLAB and CCS

a. CODE COMPOSER STUDIO

CCS provides an IDE to incorporate the software tools. CCS includes tools for code generation, such as a C compiler, an assembler, and a linker. It has graphical capabilities and supports real-time debugging. It provides an easy-to-use software tool to build and debug programs. The C compiler compiles a C source program with extension .c to produce an assembly source file with extension*.asm*. The assembler assembles an *.asm* source file to produce a machine language object file with extension *.obj*. The linker combines object files and object libraries as input to produce an executable file with extension *.out*. This executable file represents a linked common object file format (COFF), popular in Unix-based systems and adopted by several makers of digital signal processors [12,13]. This executable file can be loaded and run directly on the C6713 processor. The linear assembly source file with extension .sa, is a cross between C and assembly code. A linear optimizer optimizes this source file to create an assembly file with extension .asm (similar to the task of the C compiler).
To create an application project, one can “add” the appropriate files to the project. Compiler/linker options can readily be specified. A number of debugging features are available, including setting breakpoints and watching variables; viewing memory, registers, and mixed C and assembly code; graphing results; and monitoring execution time. One can step through a program in different ways (step into, over, or out).

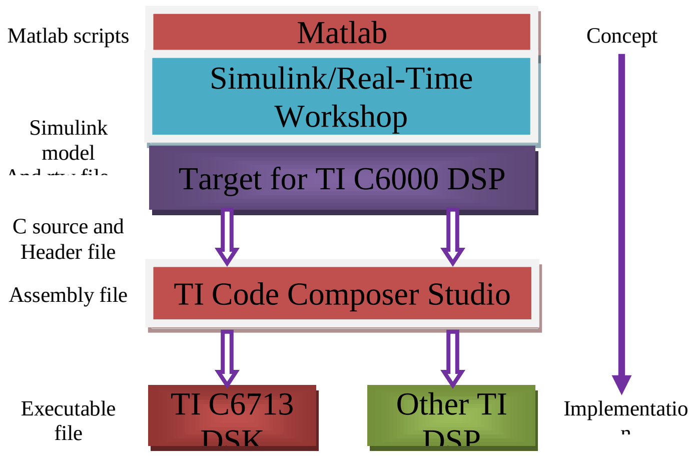


Figure 1.4 Software design flow

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 10, December 2013

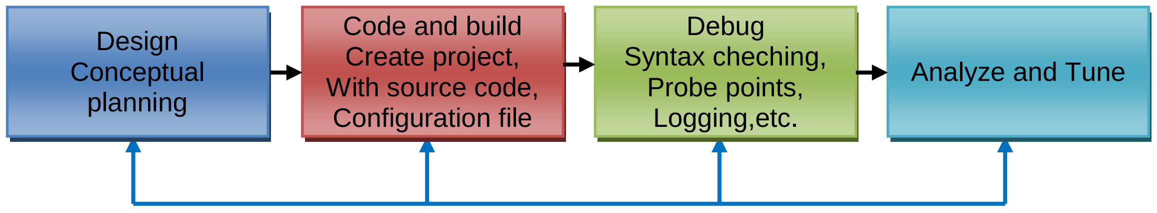


Figure 1.5 Simplified Code Composer Studio IDE Development Flow

Real-time analysis can be performed using real-time data exchange (RTDX). RTDX allows for data exchange between the host PC and the target DSK, as well as analysis in real time without stopping the target. Key statistics and performance can be monitored in real time. Through the joint team action group (JTAG), communication with on-chip emulation support occurs to control and monitor program execution. The C6713 DSK board includes a JTAG interface through the USB port.

b. SOFTWARE IMPLEMENTATION

There are several important extensions that enhance the real-time DSP programming features of MATLAB/Simulink tools. They include:

- Embedded Target for TI C6000 DSP which performs real-time prototyping and system deployment on C6000 processors.
- MATLAB Link for Code Composer which verifies and validates embedded software on Texas Instruments' C2000, C5000, C6000, OMAP, and TMS470 processors.

The main feature of these tools is that efficient working C code can be generated directly from the Simulink design as shown in figure 1.4. This greatly speeds up the development of DSP applications. Continual improvements on these tools will lead to the generation of more efficient code, in terms of memory, speed, and power consumption. The latest product, Real-Time Workshop Embedded Coder, enables users to generate, test, and deploy C code for used in real-time embedded systems. In addition, it allows hand-written C code to be added into the Simulink system for simulation and code generation for stand-alone applications.
These latest toolboxes illustrate the real-time applications as they are being designed, and are powerful research platforms for verifying the DSP algorithms. However, for generating more efficient code, fine-tuning, low-level programming, and using optimized library functions from the chip manufacturer are still needed for embedded DSP applications. Another trend in the latest tool development is the integration of MATLAB environment with the code development software from the DSP chip vendors. For example, the MATLAB Link for Code Composer allows transfer of data between MATLAB and Texas Instruments' DSP processors. Data can also be acquired from the I/O channels of the DSP board to the MATLAB for further analysis.
The model/code based design requires the following steps:

1) Simulink provides an exploratory and verification tool for both floating-point and fixed-point DSP systems and applications.
2) Simulink communicates with Code Composer Studio through Real-Time Workshop.
3) Code Composer Studio communicates with the DSK through an embedded JTAG emulator with a USB host interface.
4) Matlab communicates with CCS via Link for Code composer Studio.

Simulink uses a block based approach to algorithm design and implementation. Real-Time Workshop converts these Simulink models into ANSI C/C++ code that can be compiled using CCS. The Embedded Target for TI C6000 DSP provides the APIs required by Real-Time Workshop to generate code specifically for the C6000 platform. Link for CCS is used to invoke the code building process from within CCS to build an executable. This code can then be downloaded on the target DSP from where it runs. The data on the target is accessible in CCS or in Matlab via Link for CCS or via Real-Time Data Transfer (RTDX). This work primarily uses RTDX for accessing data on the target DSP.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

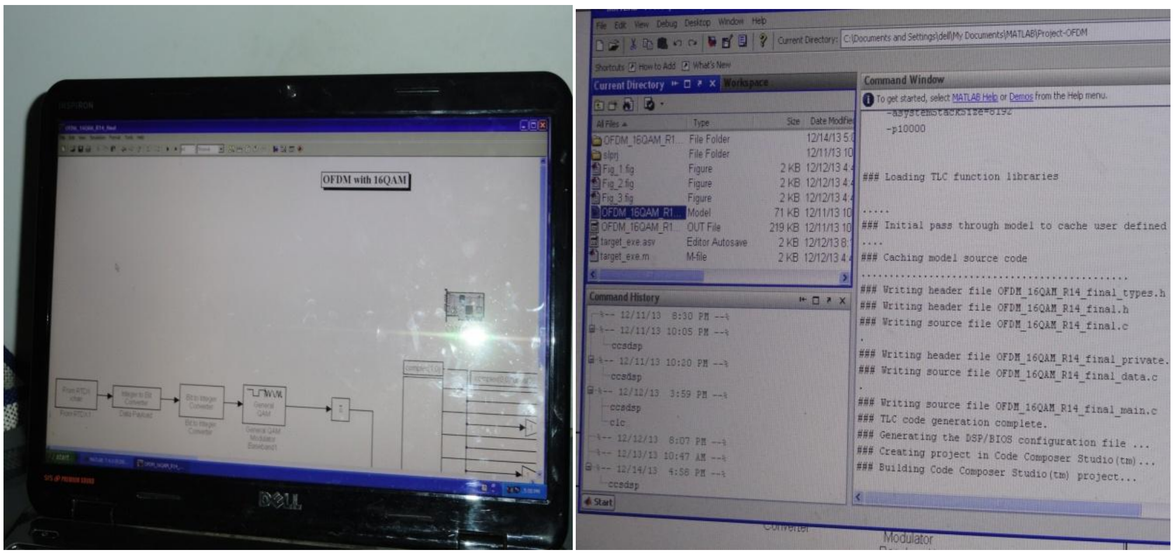
(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 10, December 2013

c. HARDWARE IMPLEMENTATION

The following steps are required for real time implementation:

1. Connect the USB cable to your PC or laptop.
2. Connect the included 5V power adapter brick to your AC power source using the AC power cord.
3. When power is applied to the board the Power on Self Test (POST) will run. LEDs 0-3 will flash. When the POST is complete all LEDs blink on and off then stay on. At this point your DSK is functional and you can now finish the USB driver install.
4. Connect the DSK to your PC or laptop using the included USB. After few seconds Windows will launch its "Add New Hardware Wizard" and prompt for the location of the DSK drivers.
5. Connect the DSK to your PC or laptop using the included USB. After few seconds Windows will launch its "Add New Hardware Wizard" and prompt for the location of the DSK drivers.
6. Follow the instructions on the screens and let Windows find the USB driver files “dsk6713.inf” and “sdusb2em.sys” on the DSK CD-ROM.
7. For testing the DSK and USB connection you can press the start button C6713 DSK Diagnostic to run the diagnostics. In approximately 30 seconds all the on-screen test indicators should turn green.
8. Now launch Code Composer Studio C6713 DSK.
9. The development flow of most DSP-based applications consists of four basic phases: application design, code creation, debug, and analysis/tuning.
10. Figure 1.4 shows basic procedures and techniques in program development flow using Code Composer Studio.



```
-p10000

### Loading TLC function libraries

.....
### Initial pass through model to cache user defined
....
### Caching model source code
............................................
### Writing header file OFDM_16QAM_R14_final_types.h
### Writing header file OFDM_16QAM_R14_final.h
### Writing source file OFDM_16QAM_R14_final.c
.
### Writing header file OFDM_16QAM_R14_final_private.
### Writing source file OFDM_16QAM_R14_final_data.c
.
### Writing source file OFDM_16QAM_R14_final_main.c
### TLC code generation complete.
### Generating the DSP/BIOS configuration file ...
### Creating project in Code Composer Studio(tm)...
### Building Code Composer Studio(tm) project...
```

Figure 1.6 Simulink and Matlab

VI. CONCLUSION

Real time implementation of OFDM transmitter and receiver in AWGN channel has been successfully implemented using MATLAB Simulink, Code Composer Studio, RTW and DSK 6713 DSP processor. Figures 1.6 show the snapshots of MATLAB and simulink and Figure 1.7 show the CCS and the set up for real time implementation i.e. laptop with Matlab, Simulink and CCS and the hardware DSK 6713 kit.

The results are given below:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 10, December 2013

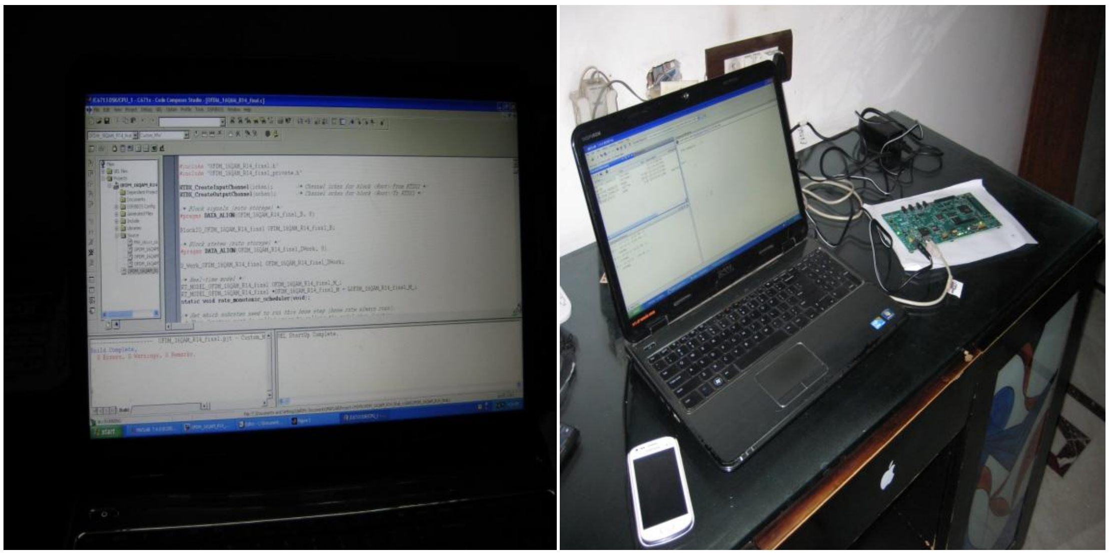


Figure 1.7 Code Composer Studio and hardware set up with DSK 6713

- Figure 1.8 shows the input/output signals and the constellation diagrams for the input/output signals with SNR=50 dB of the AWGN channel.
- Figure 1.9 shows the input/output signals and the constellation diagrams for the input/output signals with SNR=20 dB of the AWGN channel.

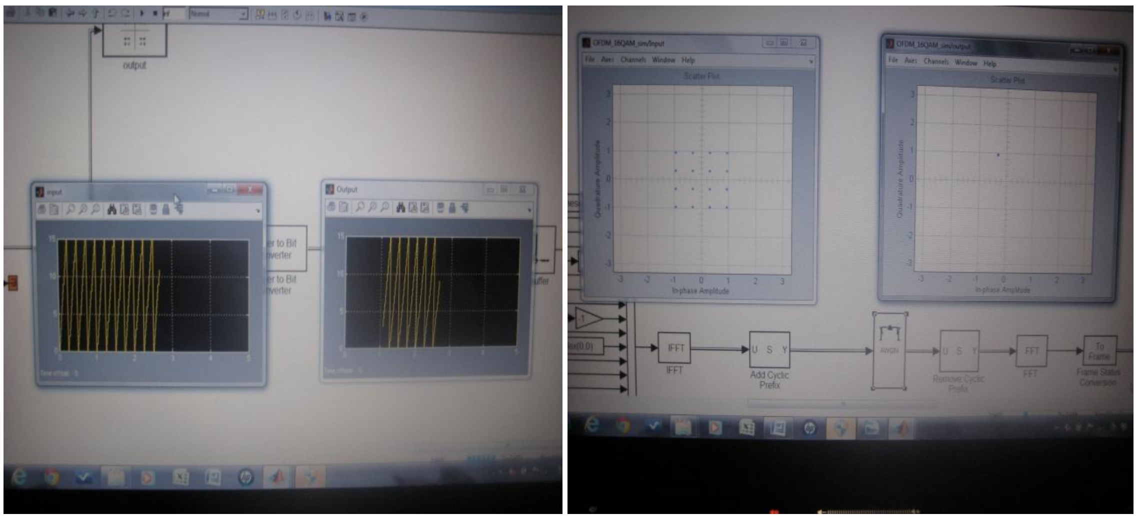


Figure 1.8 Input and output signals with their constellation diagrams with SNR=50dB

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 1, Issue 10, December 2013

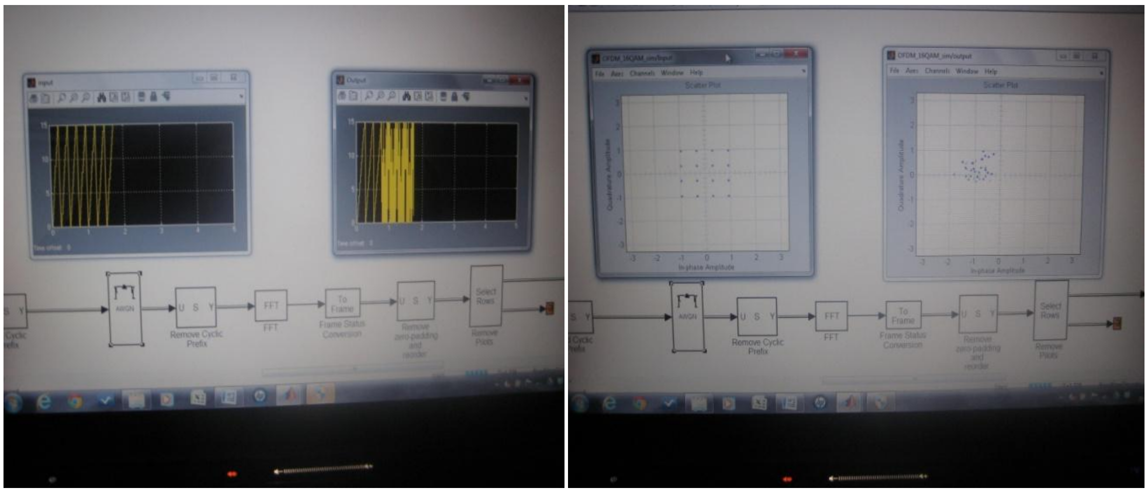


Figure 1.9 Input and output signals with their constellation diagrams with SNR=10dB

REFERENCES

1. J. Glossner, J. Moreno, M. Moudgill, J. Derby, E. Hokenek, D. Meltzer, U. Shvadron, and M. Ware, "Trends in Compilable DSP Architecture," Proc. IEEE Workshop Signal Processing Systems, pp. 181-199, 2000.
2. B.R. Wiese and J.S. Chow, "Programmable Implementations of xDSL Transceiver System," IEEE Comm. Mag., vol. 39, pp. 114-119,2000.
3. J.G. Cousin, M. Denoual, D. Saille, and O. Sentieys, "Fast ASIP Synthesis and Power Estimation for DSP Application," Proc. IEEE Workshop Signal Processing Systems, pp. 591-600, 2000.
4. TMS320C62xx User's Manual, Texas Instruments Inc., Dallas, TX, 1997.
5. W.-S. Gan, "Teaching and Learning the Hows and Whys of Real-Time Digital Signal Processing," IEEE Trans. on Education, vol. 45, no. 4, pp. 336–343,2002.
6. Jacob Fainguelernt and Arie Yeredor, "Bridging the gap between dsp theory and real-time implementation," in European DSP Education and Research Symposium (EDERS2004), Birmingham, United Kingdom, IEEE and TI,2004.
7. W.-S. Gan, Y.-K. Chong, W. Gong, and W.-T. Tan, "Rapid Prototyping System for Teaching Real-Time Digital Signal Processing," IEEE Trans. on Education, vol. 43, no. 1, pp. 19–24,2000.
8. W. S. Gan and S.M. Kuo, "Transition from Simulink to MATLAB in Real-Time Digital Signal Processing Education," The International Journal of Engineering Education, vol. 21, no. 4, Special issue on MATLAB and Simulink in Engineering Education,2005.
9. Farhan Manzoor, David Linton, Michael Loughlin,"DSP-based Implementation of a Half-Duplex Orthogonal Frequency Division Multiplexing System" ISSC 2012, NUI Maynooth, June 28-29, PP. 1-6, 2012.
10. Chien Van Trinh et.al. Implementation of a MIMO-OFDM system based on the TI C64x+ DSPICUIMC(IMCOM)'13, January 17–19, 2013, Kota Kinabalu, Malaysia,2013.
11. Gihan Gomah Hamza, Abdelhaliem A. Zekry, and Mohamed N. Moustafa," Implementation of a Complete GPS Receiver on the C6713 DSP through Simulink" Journal of Global Positioning Systems (2009) Vol.8, No.1 : 76-86,2009.
12. The Mathworks Inc., Matlab and Simulink User's Guide, 2005.
13. Texas Instruments Inc., TMS320C6416/C6713 DSK One-Day Workshop - Student Guide, 3.1 edition, Aug. 2003.