

Implementation of OFDM Based on FPGA

Zhiwei Tang⁺ and Dongqin Sheng

The third research of institute of public security, Shanghai, China

Abstract. The purpose of this paper is to introduce one hardware implementation method of Orthogonal Frequency Division Multiplexing (OFDM) using Field Programmable Gate Array (FPGA). As the FPGA hardware is reconfigurable, so the application can be developed for specific optimization of the custom structure, is ideal to achieve the required OFDM algorithms. And FPGA have become an extremely cost effective means to reduce the complexity and power consumption without sacrificing performance of OFDM system. At last this paper describes two OFDM implementation approaches, one implemented based on FPGA and the other using only an external digital signal processor (DSP). It then examines the advantages and disadvantages of these approaches from ease-of-implementation, performance, cost, perspectives and power consumption.

Keyword: OFDM; FPGA; DSP

1. Introduction

Orthogonal frequency division multiplexing (OFDM) is a multi-carrier modulation techniques, can effectively combat the frequency selective fading and narrowband interference, and higher spectrum efficiency. OFDM technology has been successfully used in asymmetric digital subscriber loop (ADSL), digital audio broadcasting (DAB), high-definition television (HDTV), Power Line Communication (PLC), wireless local area network (WLAN) and other systems. A typical OFDM Block Process is shown in Figure 1.

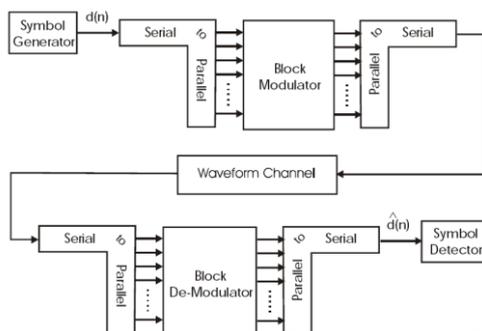


Fig. 1 OFDM Block Process

Historically, OFDM has been a relatively difficult function to implement optimally in hardware, leading many software designers to use digital signal processors in soft implementations. Unfortunately, because of the function's computationally intensive nature, such an approach typically requires multiple digital signal processors within the system to support the processing requirements. This is costly (from a device and board real estate perspective) as well as power-intensive. And OFDM is very common, computationally intensive signal processing functions found in a large number of signal processing systems. With the rapid development of microelectronic technology, FPGA devices are widely used in communication, intelligent control,

⁺ Corresponding author. Tel.: +13818375740.
E-mail address: 99chaoyang@163.com.

information processing and suchlike fields. FPGA can be suitably designed to fit virtually any function or application the user is targeting owing to the flexible nature of the FPGA's device fabric. Additionally, designers are able to customize and construct their function in a way that fully exploits the parallel nature of a hardware implementation within the FPGA, enabling better channelization, and ultimately, greater data throughput. Directing at FPGA and DSP co-processing system's characteristic, we can design an efficient realization of OFDM in order to realize real-time analysis and processing of dynamic signal. As a DSP co-processing devices, FPGA's highly parallel structure provides tremendous computing power. The following sections of this paper serve to provide an understanding of the process of implementing an OFDM modulator and demodulator algorithm in both a single digital signal processor as well as an FPGA. Subsequently, both approaches will be evaluated and analyzed from the following design considerations and perspectives: ease of implementation, cost and performance, and power consumption. This comparison employs a TMS320DM642 DSP from Texas Instruments and a Virtex FPGA from XILINX, as well as their respective design tools and software. A typical OFDM Modulator and Demodulator is shown in Figure 2.

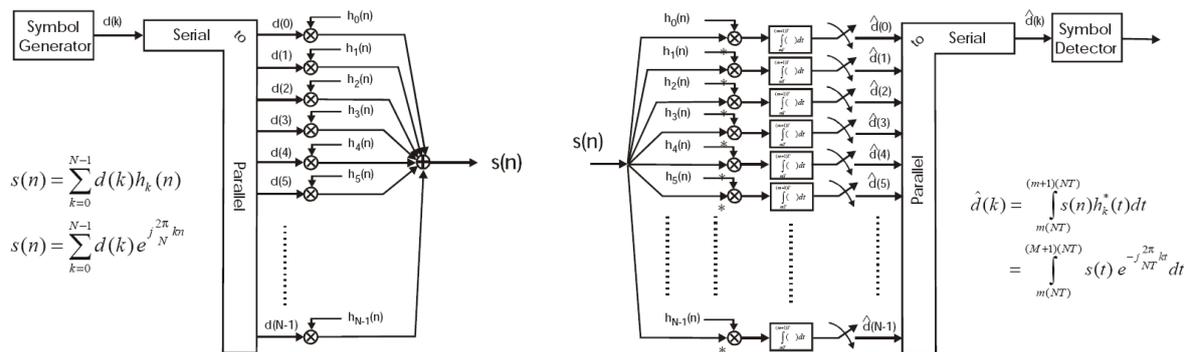


Fig. 2 OFDM Modulator and Demodulator

2. Implementing An OFDM Modulator and Demodulator Algorithm in DM642

The TMS320C64x DSPs (including the TMS320DM642 device) are the highest-performance fixed-point DSP generation in the TMS320C6000 DSP platform. The TMS320DM642 (DM642) device is based on the second-generation high-performance, advanced VelociTI very-long-instruction-word (VLIW) architecture (VelociTI.2) developed by Texas Instruments (TI), making these DSPs an excellent choice for digital media applications. The C64x is a code-compatible member of the C6000 DSP platform. With performance of up to 4800 million instructions per second (MIPS) at a clock rate of 600 MHz, the DM642 device offers cost-effective solutions to high-performance DSP programming challenges. The DM642 DSP possesses the operational flexibility of high-speed controllers and the numerical capability of array processors. The C64x DSP core processor has 64 general-purpose registers of 32-bit word length and eight highly independent functional units—two multipliers for a 32-bit result and six arithmetic logic units (ALUs).

DSP algorithms for functions like OFDM are generally supported by design tools and software that accompany the various DSP architectures. For example, Texas Instruments provides a selection of optimized, C-callable DSP library functions that is freely available for the C64x DSP family like DSP_fir_r8 and DSP_fft_16x16r for FIR filtering and OFDMs, respectively. These functions tend to run much faster than equivalent code written in ANSI C because they are hand-optimized by the processor vendor using assembly language for a targeted DSP architecture. These library functions are also changed by the vendors, thereby reducing overall system development time and allowing developers to focus on improving and differentiating their system. Typical performance for a 1,024-point 16-bit complex FFT of Modulator and Demodulator in OFDM running on a 600 MHz C64x family DSP is about 6,526 cycles or about a 15.7 μ s transform time.

3. Implementing OFDM Algorithm Based on FPGA

For FPGA, the design methodology used to synthesize each block is shown in the Fig.3.

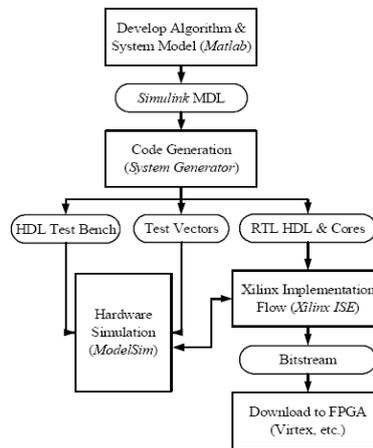


Fig. 3 Design methodology and flow diagram.

The algorithm of each block in the system is first verified by using Matlab with double precision floating point. After the algorithms are verified, the hardware implementations are obtained by constructing block diagrams in Simulink. For block and system simulations, this method enables us to integrate different components and validate their performance by exploring design trade-offs between different sets of parameters. VHDL and/or Verilog code can also be imported into Simulink via the XILINX System Generator block set, which gives flexibility to the design flow. Simulink and XILINX System Generator create bit-true and cycle-accurate hardware models which can be programmed into FPGA prototyping boards. The XILINX Integrated Software Environment (ISE) is used as the synthesizer in the design flow diagram shown below. ModelSim can also be used to verify the hardware simulation of the blocks by using test vectors generated by the System Generator or HDL test benches. Finally synthesis and performance results of the blocks are reported using ISE, and bit-streams are generated to program the FPGA boards. The following actions describe a typical flow to develop an OFDM based on FPGA.

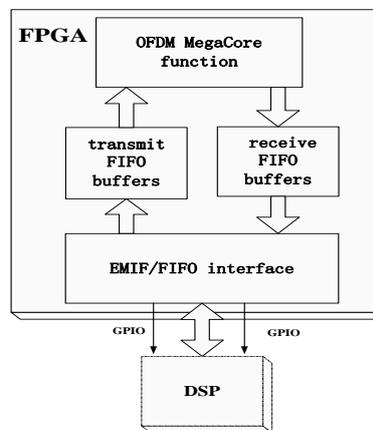


Fig. 4 Fifo used in OFDM based on FPGA

As shown in Fig. 4, The FPGA circuitry consists of four main sections: the EMIF/first in, first out (FIFO) interface, the transmit FIFO buffer, receive FIFO buffer and the OFDM MegaCore function, which performs the OFDM computation. The EMIF/FIFO interface handles the translation of the EMIF data and control signals to Atlantic™ data and control signals. All signaling between the various blocks within the FPGA is done using the Atlantic interface. The interface is a flexible interface for high-throughput packet-based data transmission of arbitrary length. It provides a synchronous point-to-point connection between two blocks of logic with flexible flow control for master-to-slave and slave-to-master directions. The flexible nature of the interface allows the designer to insert one or more co-processor module between the slave source and the Atlantic slave sink.

As storage buffers, the transmit and receive FIFO buffers handle the transfer of the data packets between the EMIF and the OFDM MegaCore. For a 1,024-point FFT of OFDM, the transmit and receive FIFO buffers

are set to a depth of 2,048 to avoid any overflow of data that might occur when writing to the FIFOs. The OFDM MegaCore function is an optimized, parameterizable IP block available from XILINX. The EMIF was chosen as the connection medium for the FPGA due to the data transfer rates available and the possibility of using the enhanced direct memory access (EDMA) controller integrated within the DSP. Table 1 shows the peak data-transfer rates based on the given EMIF clock rates. In practice, lower overall rates will be achieved, e.g., when transferring data between two resources that share the EMIF.

Table 1 EMIF Peak Data Rates

EMIF data- band&mode	Peak transfer(Mbps)		
	66MHz	100MHz	133MHz
32bit asynchronous	56	85	100
32bit synchronous	278	420	550
64bit synchronous	565	850	1125

The DMA controller within the TMS320DM642 DSP transmits packets of data to be processed via a synchronous EMIF to the EMIF/FIFO interface on the Virtex FPGA. The EMIF/FIFO interface translates the EMIF signals to Atlantic-based signaling so that appropriate data packets can be loaded into the transmit FIFO buffer. The control logic surrounding the transmit FIFO buffer monitors the fill level of the transmit FIFO to determine when a new input data packet of 1024 samples is available for processing by the OFDM function. The transmit FIFO can start sending data across the Atlantic interface to the OFDM when the transmit FIFO buffer is almost full.

The output of the OFDM is buffered by the receive FIFO buffer to allow for periods when the EMIF is busy. A DMA transfer is requested when a whole packet of processed data is available to be read from the receive FIFO buffer by the DSP. This occurs when they receive FIFO buffer is almost full. The software that was implemented with the OFDM coprocessor design was built using the DSP functions and BIOS libraries included with the Texas Instruments Code Composer Studio software to configure the EDMA controller and interrupts. The software was used to stream blocks of data, generated using a sine wave generator algorithm, through the OFDM co-processor. The sine wave generator is implemented using a double precision cordic algorithm on the digital signal processor. The output values can be graphed within the Code Composer Studio workspace for visual verification.

Two DSP general-purpose I/O (GPIO) pins are dedicated for use as event triggers to the EDMA: one for transmitting data from the digital signal processor to the co-processor and one for receiving data from the co-processor to the EDMA on the digital signal processing. The co-processor requests a new transmit DMA whenever the OFDM function is free and when the transmit credit register (TX_CREDIT) is non-zero. The digital signal processor must write to TX_CREDIT before the co-processor can begin operating. This way, the digital signal processor can signal to the coprocessor that the data buffers are ready.

When the OFDM has completed processing the data and the data is available in the receive FIFO buffer, the coprocessor control logic requests a receive DMA from the digital signal processor. Each time a DMA is completed, the EDMA sends an interrupt request to the digital signal processor. The software tracks the number of packets transmitted and received. When a pre-defined number of packets have completed processing, the software calculates the average performance of the OFDM co-processor across packets. The software that was created performs the following tasks within the main () routine.

4. Considerations

The following subsections attempt to evaluate and analyze the advantages and disadvantages of both methods from an ease of implementation, cost and performance, and power consumption perspective. These are some of the main factors designers consider when deciding on the best implementation for their systems.

4.1. Ease of Implementation–Evaluation & Analysis

For OFDM, the DSP-only approach significantly reduces the implementation effort with the availability of pre-built, assembly optimized, C-callable library functions. Even if the designer requires a custom function that is not available with generic library functions, complex algorithms are generally easier to implement in a high-level language like C or C++. Challenges may come while trying to optimize the performance of the function for a particular DSP, often requiring an in-depth knowledge of the processor architecture and assembly instructions. Nevertheless, the designer remains within the same familiar development environment, and is not required to build additional hardware functions to complement the desired system.

On the other hand, the FPGA co-processor approach currently requires a certain amount of hardware knowledge to assemble the various components of the FPGA co-processing system, consisting of the EMIF/FIFO interface, transmit and receive FIFO buffers, and the actual co-processing function itself. The transmit and receive FIFO buffer sizes will have to be individually parameterized to suit each co-processor within the system. The availability of parameter architecturally optimized pre-built IP functions from FPGA vendors like XILINX, however, in the implementation of the OFDM, reducing the overall design and verification time of the system. With the FPGA co-processor approach, familiarity with DSP software development environments, as well as hardware design methods, is necessary to successfully integrate both elements of the system. Despite this apparent hurdle, the future of FPGA co-processing looks bright with the continuous evolution of better and better system integration tools like XILINX SOPC Builder which may eventually support built-in interfaces to DSPs like Texas Instruments' EMIF and Analog Devices' Link Port interface.

4.2. Cost and Performance–Evaluation & Analysis

It was established that a 600 MHz TMS320DM642 DSP is capable of completing a 1,024-point 16-bit fixed-point complex FFT of OFDM in about 6,526 cycles or about 15.7 μ s. A similar OFDM configuration running on a Virtex FPGA is capable of achieving transform times of up to 7.97 μ s (1,291 clock cycles at 266 MHz), while consuming about 15 percent of the entire FPGA. The FPGA co-processor example used here is capable of completing a 1,024-point, 16-bit complex OFDM in about 11.6 μ s at 133 MHz on a 64-bit synchronous EMIF configuration. Overall, the FPGA coprocessor shows a 16.5 times improvement from a relative price/performance perspective.

Running the EMIF at higher clock rates (133 MHz rather than 66 MHz) or at a higher bandwidth (64-bit synchronous rather than 32-bit synchronous) could potentially increase the performance of the FPGA co-processor assuming that the performance bottleneck is caused by the latency in data transfer. The use of FIFO buffers in the transmit-and-receive paths enables the OFDM (or any other co-processing function) to run at higher clock speeds from the EMIF. This is useful for system optimization if the co-processor function is found to be the performance bottleneck. The co-processor function can then be run at a higher speed compared to the EMIF to decrease the processing time.

4.3. Power Consumption – Evaluation and Analysis

Power consumption on a digital signal processor and FPGA is a function of the underlying CMOS process used to fabricate the devices. The basic formula used to evaluate power for a CMOS circuit is Power, P (W) = $1/2 * C * V^2 * f$, where V is the supply voltage in volts; C is the load capacitance in Farad; f is the operational clock frequency in Hertz. From this formula it is easy to see that by reducing any one of the three main components, capacitance, voltage or operational frequency, we are able to reduce the overall power consumed by the device.

Power consumption in a digital signal processor is attributed to a few or all of the following factors: the type of instructions being implemented by the processor (which affects the switching from a logic 0 to a logic 1 and vice versa on the device, which in turn, affects power consumption, since the more frequently switching occurs, the larger the capacitance on a signal path, causing more power to be consumed), the clock frequency at which the processor is running, the operational voltage of the processor, and the number of peripherals being used or enabled . The power consumption reported for the C64x DSP family running at 600 MHz for

typical activity is approximately 1.2 Watts. Based on the information provided in, this estimate can and will vary depending on the type of application being performed on the digital signal processor.

Not unlike a digital signal processor, the power consumed by an FPGA is also subject to similar factors such as operating voltage, operational clock frequency, and the number of logic 0 to logic 1 and logic 1 to logic 0 crossings. Along with operational voltage, the FPGA also has an additional source of power consumption known as leakage or core power. This is the amount of power required to “turn on” or enable the underlying FPGA fabric and configuration memory to maintain proper FPGA operation. This core power consumption occurs whether or not the FPGA has been configured and is performing its programmed tasks. The power consumed by the operation of an application on the FPGA is subject to the same factors and conditions as a DSP or any other application running on a CMOS-based technology.

The ISE software by XILINX that supports the Virtex device family has a built-in power estimator that reports the approximate power consumed by a particular application for a given set of stimuli. The stimuli or vectors used for this power estimation are important, since the toggle rate of data is a key component in power consumption. The designer has to be careful that the vectors used in the estimation are indicative of the signals that will be passing through the system during actual operation to get more accurate representations of the power that will be consumed in the field. Approximate power estimates for a 1,024-point 16-bit complex OFDM built using the OFDM MegaCore function was obtained using the ISE software. Table 2 shows the power estimation numbers for operating clock frequencies of 66 MHz, 133 MHz and 266 MHz.

Table 2 Power Consumption at different Emif Hz

OFDM MegaCore Function	Power Consumption(w)		
	66MHz	133 MHz	266MHz
	0.7893	0.8903	1.186

From a function to function comparison, we can see that an FPGA OFDM implementation generally utilizes less power than a similar OFDM function implemented on a DSP. Even at the maximum operating clock frequency of 266 MHz, the FPGA OFDM implementation still consumes less power than the digital signal processor. In this case, the FPGA is not fully utilized (19 percent in this case) and can handle more functions and larger amounts of data in parallel compared to the digital signal processor. Increased functionality added to the FPGA will most likely increase the toggle rate, the operating frequency and the amount of logic utilized within the FPGA, all of which will increase the total amount of power consumed by the FPGA. In return, the designer gets the benefit of a higher performance system, a trade-off the system designer should take into account. Additionally, with the FPGA co-processor approach, the designer has to factor in the addition power consumed by the digital signal processor (now running in a reduced function, because a large portion of the processing has been off-loaded to the FPGA), in addition to the power consumed by the FPGA co-processor.

5. Conclusions

DSP and FPGA solutions provide designers with a myriad of implementation options and solutions for today’s system designers. Along with these solutions comes a variety of design factors and considerations that need to be evaluated to select the best approach, depending on system requirements like ease of implementation, cost and performance as well as power consumption.

OFDM communication system, which requires many OFDM operations, is to be computed for a large number of channels. Today’s digital signal processors are unable to keep up with the load required of these systems unless an approach requiring large arrays of digital signal processors is employed. Integration of these digital signal processors within the system is not a trivial task, as the coordination of shared memory between processors is especially complex.

We showed that FPGA device has sufficient resources to accommodate the entire OFDM system. FPGA is capable of parallelizing the operation of these functions, reducing the overall computation time of each

operation and are able to support a larger number of channels within a single device. Additionally, the densities of FPGAs have grown significantly over the last two years to the point that multiple instantiations of these functions can be implemented within a single FPGA, reducing the total number of devices required and ultimately board real estate. From a price/performance comparison, FPGA provide better performance for lower cost compared to a single digital signal processor approach. Additionally, because the FPGA is not fully utilized, more functionality and parallelism could be added to the FPGA to increase the amount of processing the FPGA is capable of without impacting the cost of the system. Also, from a function-to-function power comparison, we see that for the same function, an FPGA implementation is capable of consuming less power than a digital signal processor. For comparable performance to an FPGA, a designer may be required to implement an array of multiple digital signal processors, something which could possibly increase the cost and power consumption of a system beyond that of an OFDM implementation based on FPGA.

6. Acknowledgment

The authors wish to acknowledge, and to thank, the sponsors of this work: DSP&FPGA Systems of Wireless Communication Process.

7. References

- [1] M. Grajcar and B. Sick, "The OFDM butterfly operation in 4 processor cycles on a 24 bit fixed-point DSP with a pipelined multiplier," ICASSP 97, vol.1, pp. 611-614, Munich, Germany, April 1997.
- [2] S. Hsiao and C. Yen, "New unified VLSI architectures for computing DFT and other transforms," ICASSP 97, vol.1, pp. 615-618, Munich, Germany, April 1997
- [3] S. Scalera, M. Falco, and B. Nelson, "A Reconfigurable Computing Architecture for Microsensors," Proceedings, Field-Programmable Custom Computing Machines (FCCM), Napa, CA, April 17-19, 2000, pp. 59-67.
- [4] H. Shekarforoush and R. Chellappa, "A Multifractal Formalism for Stabilization and Activity Detection in FLIR Sequences," Proceedings, ARL Federated Laboratory 4th Annual Symposium, College Park, MD, March 2000, pp. 305 – 309.
- [5] N. Vaswani and R. Chellappa, "Object Detection and Compression Techniques for FLIR Sequences," Proceedings, ARL Federated Laboratory 5th Annual Symposium, College Park, MD, March 2001, pp. 189 – 194.
- [6] J. E. Scalera, "Image Chipping with a Common Architecture for Microsensors", M.S. Thesis, Bradley Dept. of Electrical and Computer Engineering, Virginia Tech, 2001.
- [7] S. Lim, and P. Ekas, "Design Methodology for Hardware Acceleration for DSP", Proc. International Signal Processing Conference, GSPx, 2003.
- [8] H. Kwan et al, "Three-dimensional OFDMs on a digital-signal parallel processor with no interprocessor communication," 30th Asilomar Conf. on Signals, Systems and Computers, Pacific Grove, CA, Nov. 1996.
- [9] OFDM MegaCore Function User Guide, version 2.1.0, XILINXa Corporation, June 2004.