CONFIGURABLE, RESOURCE-OPTIMIZED FFT ARCHITECTURE FOR OFDM COMMUNICATION

Inkeun Cho[1]*, Chung-Ching Shen*[1]*, Yahia Tachwali*[2]*, Chia-Jui Hsu*[2] *and Shuvra S. Bhattacharyya*[1]

[1]Department of Electrical and Computer Engineering, and
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20740, USA
{inkeun, ccshen, ssb}@umd.edu
[2]Agilent Technologies Incorporation
Westlake Village, CA 91362, USA
{yahia_tachwali, jerry_hsu}@agilent.com

ABSTRACT

In this paper, we present a designer-configurable, resource efficient FPGA architecture for OFDM system implementation. Our design achieves a significant improvement in resource efficiency for a given data rate. This efficiency improvement is achieved through careful analysis of how FFT computation is performed within the context of OFDM systems, and streamlining memory management and control logic based on this analysis. In particular, our OFDM-targeted FFT design eliminates redundant buffer memory, and simplifies control logic to save FPGA resources. We have synthesized and tested our design using the Xilinx ISE 13.4 synthesis tool, and compared the results with the Xilinx FFT v7.1, which is a widely used commercial FPGA IP core. We have demonstrated that our design provides at least 8.8% enhancement in terms of resource efficiency compared to Xilinx FFT v7.1 when it is embedded within the same OFDM configuration.

Index Terms— OFDM, FPGA, Resource, FFT.

1. INTRODUCTION

Orthogonal frequency division multiplexing (*OFDM*) is a multiple access scheme for high data rate communication. OFDM is based on frequency division multiplexing, where bandwidth is divided into a series of non-overlapping frequency bands, and each user is assigned a dedicated subset of the available bands. OFDM is a special case of frequency division multiplexing in which the sub-carriers are orthogonal to one another, which eliminates crosstalk between sub-carriers [1].

OFDM offers a variety of advantages, including robustness against narrow band co-channel interference, inter symbol interference and fading caused by multipath propagation. Also, OFDM can readily be implemented with the fast Fourier transform (*FFT*). Due to these advantages, OFDM is widely used in modern wireless communication systems, including wireless LAN, WiMAX and 3G LTE. Figure 1 shows a system overview for OFDM.

As OFDM is widely used in wireless communication system, efficient hardware implementation for OFDM is critical issue. FFT computation is the most computationally intensive part of an OFDM system. A large body of prior research exists on efficient implementation of FFTs. Two general forms of FFT architecture are the parallel input and output architecture [2] [3] [4], and the serial input and output architecture [5] [6].

The parallel input and output architecture can offer relatively high throughput of computation, however it requires multiple computation units (`butterfly` processing units) for processing parallel of inputs simultaneously, and additional hardware resources due to complex multipliers [7]. Since area efficiency is critical in mobile systems, this architecture is not well suited for OFDM implementation. On the other hand, the serial input and output architecture can be developed using single computation units along with random access memory (*RAM*), and can thereby offer resource efficient design for OFDM implementation.

Saeed et al. presented an FFT system in [6] that employs a simple radix-2 based processing unit instead of a radix-4 unit, and provides a configurable FFT size (number of points). This system also uses simple counter based control logic, which helps to reduce resource utilization. Our proposed solution adopts ideas from the work of Saeed et al., and further streamlines buffer memory and control logic implementation based on the context in the which an FFT unit operates within an overall OFDM system.

A variety of methods have been presented for implementing serial input and output FFTs for OFDM systems. Hung, Chen and Chen presented an address generator for memory and twiddle factors of variable-length FFTs for high speed and low complexity design [8]. Jung, Yoon and Kim demonstrated a method for efficient design of the radix-4 based butterfly [9] and multiplier in an FFT unit. This approach was shown to achieve enhancements in both area and throughput. Huang et al. presented a memory efficient OFDM system considering WiMAX subcarrier allocation scheduling and demonstrated results in terms of area efficiency [10]. However, these approaches are based on specialized OFDM system requirements — i.e., these methods exploit the specialized structure for specific OFDM system configurations to derive optimized designs.

In contrast, in this paper we target efficient FFT implementation for general OFDM architectures, which can be configured across a range of different specialized OFDM requirements. Such general OFDM architectures are useful for prototyping as well as for cognitive radio applications, which may support many different communication standards within the same hardware.

For such generalized OFDM architecture design, we have identified inefficiencies in memory and resource utilization in conventional FFT implementations, and we have developed methods to address these inefficiencies to achieve a novel area-efficient FFT hard-

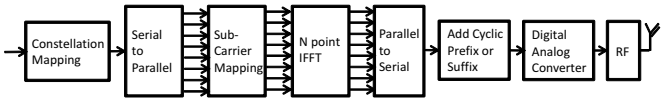

Fig. 1. OFDM system overview.

ware architecture. To help support the overall context within a generalized OFDM design, our design provides significant user configurability in terms of FFT parameters, such as the number of FFT points, and fixed point representation format.

To validate our FFT architecture, we have synthesized it using Xilinx ISE version 13.4 with the Digilent Spartan 3e platform as the target. This target platform is based in turn on the Xilinx (`xc3s500efg320-4`) integrated circuit. We have selected the Xilinx FFT intellectual property (IP) core (Xilinx FFT IP core v7.1) as a reference for comparison, as it is a widely used commercial IP core.

2. FFT DESIGN OPTIMIZATION FOR OFDM

2.1. FFT Design with User Configurability

OFDM modules are used in many communication systems, and OFDM configurations in general need to be varied with communication system requirements. To support a broad randge of OFDM configurations, we consider a parameterized OFDM system design that supports flexibility across the OFDM design space. A block diagram of our targeted OFDM system design is shown in Figure 2. The contribution of this paper is to present an optimized FFT architecture that is suitable for integration as a component of this system design.

There were plenty of FFT designs were researched and the FFT implementation is a well matured research area. In our paper, we focused more on developing the efficient OFDM modules based on the FFT module which was presented in previous research. We chose the FPGA FFT design in [6] among plenty of system, because the system can easily expandable with the point of FFT for supporting multiple configuration and have a counter based control logic. The proposed configurable parameters for OFDM module provides user configurability in terms of parameters such as the number of FFT points, and fixed point representation format.

2.2. Memory Efficient Design of FFT for OFDM

In an OFDM system, a guard interval using a cyclic prefix or cyclic suffix is used between consecutive OFDM symbol durations [1]. Such prefix or suffix based guard intervals are used to combat inter symbol interference. When using a cyclic prefix, the OFDM output symbol size increases from N to $(N+C)$, where N and C represent the FFT size and cyclic prefix size, respectively. The OFDM symbol duration is the sum of the actual transmission time for an N-sized IFFT output and the guard interval time. The actual transmission time for an OFDM symbol is determined by the time taken by the physical layer electronics, and the RF transmission time. If the rate at which OFDM symbols are generated exceeds the throughput of the physical layer electronics, an additional FIFO is inserted to help compensate for the rate difference.

In OFDM system design, a buffer of size $(N + C)$ is inserted to handle cyclic prefix or suffix generation. In our design, we employ a cyclic suffix, and configure the FFT a with `stall_state`. In this state, which is employed during guard intervals, the FFT does not generate any output. After FFT data is generated, a dedicated `cyclic_suffix` module suffixes data in memory based on a suffix size parameter `C`. This approach allows for streaming generation of OFDM symbols before digital-to-analog conversion without requiring additional buffering, and provides a memory savings of

$$M_{save1} = \frac{N}{N+C}. \quad (1)$$

Within an OFDM system, a re-order buffer is typically used in conjunction with the FFT unit. The output streams for the IFFT/FFT computations are in bit-reversed order based on common conventions for IFFT and FFT implementation. A common buffer management scheme for re-order buffer and cyclic suffix design, which is presented in [11], uses a buffer of size $2N$ for re-ordering and continuous processing of FFT output data. However, with optimized buffer management, our design employs a smaller re-order buffer of size $(N + C)$. In particular, based on knowledge of how the FFT is employed in the surrounding OFDM context, we are able to replace the double buffering scheme of [11] with in-place buffering. This leads to a re-order buffer that is smaller by a factor of

$$M_{save2} = \frac{N-C}{2N}. \quad (2)$$

This represents a significant savings since typically C is much smaller than N in OFDM systems.

2.3. Simple Control Logic

Another opportunity that we exploit in our OFDM-oriented FFT subsystem is the ability to share control logic across key modules that interface to the FFT. In particular, we extend the controller for the core FFT unit with an additional counter, and use the resulting circuit as a unified controller for the re-order buffer and cyclic suffix modules ("interface modules"). This is in contrast to conventional approaches to OFDM system implementation that use off-the-shelf or otherwise generic FFT IP blocks and interface modules, which come with their own controllers. Our approach to sharing of control logic helps to further enhance the area efficiency of our approach, as we demonstrate quantitatively in Section 3.

3. EXPERIMENTS

3.1. Evaluation Metric

In this section, we demonstrate the performance and area efficiency of our OFDM-oriented FFT subsystem design. First, we define two evaluation metrics that we employ in our experiments. As a measure of resource utilization on the targeted FPGA devices, we define $R(D_i)$ to be the number of FPGA slices occupied by the synthesized result for a given design D_i. Also, given two designs D_i and D_j such that $R(D_j) > R(D_i)$, we use the metric $E_(D_i, D_j)$ to quantify the degree to which D_i is more resource-efficient compared to D_j. This metric is defined by:

$$E_R(D_i, D_j) = (1 - \frac{R(D_i)}{R(D_j)}) \times 100(\%). \quad (3)$$

3.2. Performance of Our Proposed Design

To further test the performance of our proposed design, the design is synthesized under Xilinx ISE version 13.4 with the option of using the Digilent Spartan 3e platform, which is based in turn on the Xilinx integrated circuit. We have chosen FFT computation with cyclic

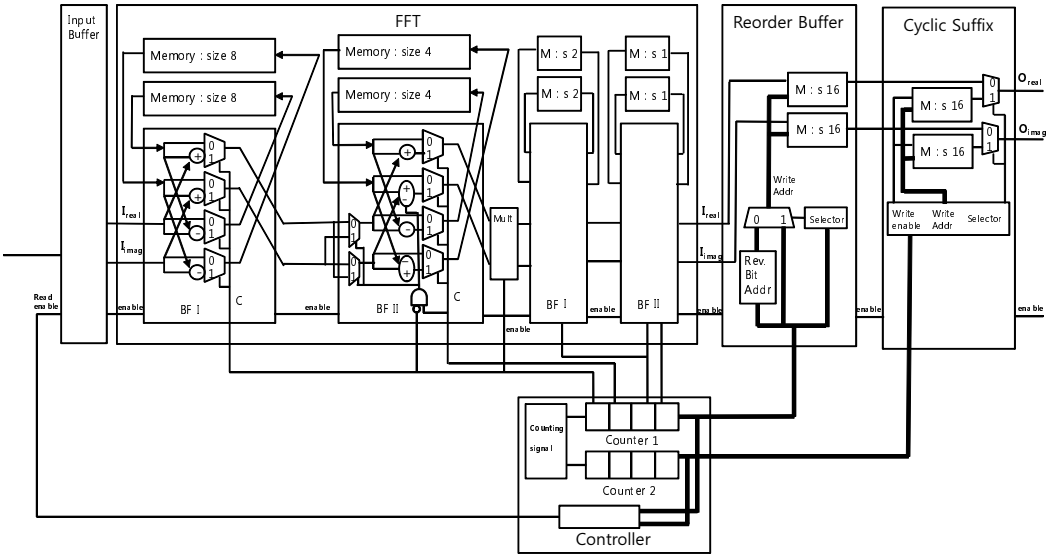

Fig. 2. Block diagram of targeted OFDM system design.

Xilinx FFT Configuration Parameter	Applied Value
Target Frequency	Same as our design
Hardware Architecture	Pipelined Architecture
I/O Method	Streaming I/O
Data Format	12 Bit Fixed Point Representation
Length of Precision	Same as our design
Output Ordering (Bit Rev. or Natural)	Same as our design
Number of Stages Using Block RAM	Same as our design
Complex Multiplier	3-multiplier structure (resource optimization)

Fig. 3. Configuration for Xilinx FFT v7.1 IP Core Generation

	FFT			FFT with Natural Order Output			FFT with Natural Order Output and Cyclic Suffix		
	R	M	Max Speed	R	M	Max Speed	R	M	Max Speed
16 Points FFT	616 / 4,656 13%	4 / 20 20%	25.333MHz	659 / 4,656 14%	6 / 20 30%	25.333MHz	698 / 4,656 14%	8 / 20 40%	25.296MHz
256 Points FFT	1,615 / 4,656 34%	12 / 20 60%	10.279MHz	1,666 / 4,656 35%	14 / 20 70%	10.292MHz	1,703 / 4,656 36%	16 / 20 80%	10.279MHz
1024 Points FFT	2,349 / 4,656 50%	16 / 20 80%	7.819MHz	2,405 / 4,656 51%	18 / 20 90%	7.819MHz	2,450 / 4,656 52%	20 / 20 100%	7.819MHz

Fig. 4. Performance of our proposed configurable FFT design.

	FFT		FFT with Natural Order Output		FFT with Natural Order Output and Cyclic Prefix	
	R	M	R	M	R	M
16 Points FFT	784 / 4,656 16%	0 / 20 0%	897 / 4,656 19%	1 / 20 5%	1,010 / 4,656 21%	1 / 20 5%
256 Points FFT	1,965 / 4,656 42%	4 / 20 20%	2,032 / 4,656 43%	5 / 20 25%	2,160 / 4,656 46%	5 / 20 25%
1024 Points FFT	2,608 / 4,656 56%	7 / 20 35%	2,687 / 4,656 57%	9 / 20 45%	2,848 / 4,656 61%	9 / 20 45%

Fig. 5. Performance of Xilinx FFT core.

suffix generation as the configuration for the evaluating system performance. The FFT module is the most computationally complex part in an OFDM system and the system resource usage is heavily related to this part. Also, the Xilinx IP generator supports FFT v7.1 instantiation along with features for reorder buffer and cyclic suffix usage. Since this core (the Xilinx FFT IP core v7.1) has these relevant features and it is a widely used commercial core, we employ it as a reference for comparison in our experiments.

Results obtained from our synthesis experiments are shown in Figure 4. In this Figure, R represents the amount of occupied logic resources, and M represents the amount of RAM memory blocks that are used. Each entry for R is expressed in the form x/y, followed by a percentage value z. Here, x represents the number of occupied logic resources in the synthesized design, y represents the total number of available logic resources on the targeted FPGA device, and z represents the percentage of occupied resources (i.e., the quotient (x/y)).

For fairness in our comparison, we maintained consistency where possible between the configurations used for our design and the configurations used for generating the Xilinx FFT core. Figure 3 shows the configurations that we used to generate the Xilinx FFT core.

For the synthesis experiments using the Xilinx core, the target clock frequency is set based on the maximum frequency at which our proposed configurable FFT module can operate. Since the Xilinx core generator generally improves the resource utilization efficiency for lower speeds, this optimizes the resource utilization of the Xilinx core for the same speed that is achieved by proposed design.

For the complex multiplier configuration in the Xilinx IP core, we applied the `3-multiplier structure` among the three available options — `CLB-logic`, `3-multiplier structure` and `4-multiplier structure`. We selected the `3-multiplier structure` because it is the most resource-efficient option.

Figure 6 shows the resource efficiency of our proposed design over the reference design, based on the experiments summarized in

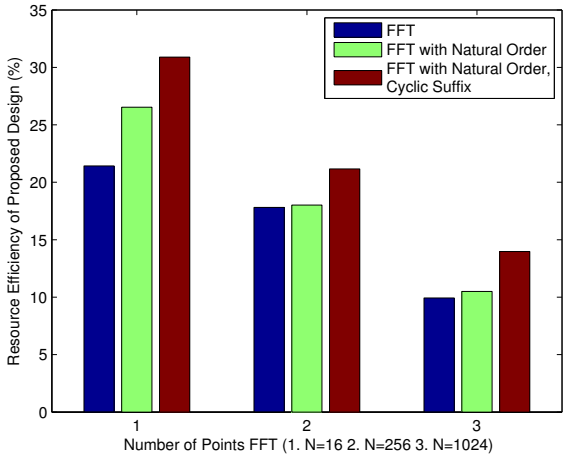

Fig. 6. Resource efficiency of proposed design with block RAM.

	Num of Points	FFT	FFT with Natural Order Output	FFT with Natural Order Output and Cyclic Prefix
Proposed Design	16 Points FFT	608 / 4,656 13%	682 / 4,656 14%	737 / 4,656 15%
	256 Points FFT	1,884 / 4,656 40%	2,476 / 4,656 53%	3,016 / 4,656 64%
Xilinx FFT Core	16 Points FFT	784 / 4,656 16%	1,198 / 4,656 25%	1,247 / 4,656 26%
	256 Points FFT	2,067 / 4,656 44%	3,724 / 4,656 79%	3,847 / 4,656 82%

Fig. 7. Performance comparison of our proposed design based on a distributed memory implementation.

Figure 4 and Figure 5. We see that our proposed configurable FFT module requires less logic resources than the Xilinx FFT core. Thus, our proposed module is advantageous for highly cost-constrained design scenarios with extensive requirements of computational resources, such as those that arise from trying to fit complex wireless communication subsystems into single-chip implementations.

Although, as shown in Figure 4 and Figure 5, our design is significantly more efficient in terms of logic resource requirements, it has increased requirements in terms of on-chip memory blocks.

An alternative to incurring the increased block RAM cost of our approach, is to employ distributed memory, which consumes logic resources (slices) rather than block RAM. To study the impact on resource efficiency in this case, we synthesized the design using distributed instead block RAM, and and we compared the resulting resource usage with the Xilinx FFT core under the same conditions (i.e., by enforcing use of distributed RAM). Figure 7 shows that under these constraints, our proposed configurable FFT module requires less logic resources than the Xilinx FFT core.

Figure 8 shows the resource efficiency (E_R) for the distributed memory version of our proposed design compared to the Xilinx FFT core. This demonstrates an efficiency enhancement of 8.5% using our proposed OFDM-oriented FFT solution.

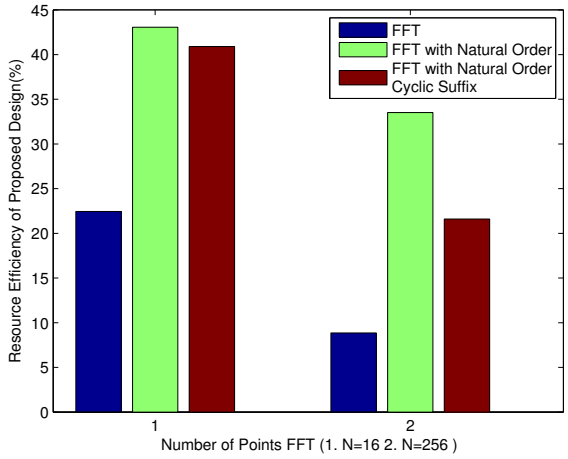

Fig. 8. Resource efficiency of our proposed design using distributed memory.

4. CONCLUSION

In this paper, we have presented a resource efficient design for a configurable FFT module for flexible integration into OFDM systems. Our design overcomes inefficiencies in the conventional use of FFT modules in OFDM systems, and provides extensive designer-configurability that helps to support a variety of communication system specifications. Our OFDM-targeted FFT design achieves efficient resource utilization through careful buffer memory optimization, and streamlining of control logic. To validate our proposed FFT design, we synthesized it on an FPGA target, and compared the synthesis results to an analogous configuration of a commercial, off-the-shelf FFT core. Results from synthesis show that our proposed design provides significant improvements in resource efficiency under both block RAM and distributed memory based implementations.

5. REFERENCES

[1] H. Rohling, Ed., *OFDM: Concepts for Future Communication Systems*, Springer, 2011.

[2] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," *IEEE Transactions on Computers*, vol. C-33, no. 5, pp. 414–426, 1984.

[3] W. Han, T. S. Arslan, A. T. Erdogan, and M. M. Hasan, "Multiplier-less based parallel-pipelined FFT architectures for wireless communication applications," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, 2005, pp. v/45–v/48.

[4] J. Palmer and B. Nelson, "A parallel FFT architecture for FPGAs," in *Field Programmable Logic and Application*, J. Becker and S. Vernalde M. Platzner, Eds., vol. 3203 of *Lecture Notes in Computer Science*, pp. 948–953. Springer Berlin Heidelberg, 2004.

[5] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proceedings of the International Parallel Processing Symposium*, 1996, pp. 766–770.

[6] A. Saeed, M. Elbably, G. Abdelfadeel, and M. I. Eladawy, "Efficient FPGA implementation of FFT/IFFT processor," *Inter-*

national Journal of Circuits, Systems and Signal Processing, vol. 3, no. 3, pp. 103–110, 2009.

[7] H. Jiang, H. Luo, J. Tian, and W. Song, "Design of an efficient FFT processor for OFDM systems," 2005, vol. 51, pp. 1099–1103.

[8] C.-P. Hung, S.-G. Chen, and K.-L. Chen, "Design of an efficient variable-length FFT processor," in *Proceedings of the International Symposium on Circuits and Systems*, 2004, pp. II–833–II–836.

[9] Y. H. Jung, H. I. Yoon, and J.-S. Kim, "New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 1, pp. 14–20, 2003.

[10] S.-J. Huang and S.-G. Chen, "A memory-efficient continuous-flow FFT processor for Wimax application," in *Proceedings of the International Symposium on Circuits and Systems*, 2012, pp. 17–20.

[11] F. Kristensen, P. Nilsson, and A. Olsson, "Reduced transceiver-delay for OFDM systems," in *Proceedings of the Vehicular Technology Conference*, 2004, vol. 3, pp. 1242–1245.