Algorithmic Optimizations for Energy Efficient Throughput-oriented FFT Architectures on FPGA

Ren Chen and Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, USA 90089
Email: {renchen, prasanna}@usc.edu

***Abstract*—Energy efficiency is a key design metric when implementing signal processing applications on FPGAs. In this paper, high level energy optimizations are proposed to facilitate the development of an energy efficient throughput-oriented FFT design. At the algorithm mapping level, we develop a data remapping technique and a memory activation scheduling method to reduce memory energy consumption. At the architecture binding level, we explore and identify the optimal memory binding scheme and pipelining strategy of the floating point units. The experimental results show that the dynamic power dissipation is reduced significantly using the proposed algorithmic optimizations. Compared with the baseline architecture, the optimized architecture achieves 2.97x, 2.99x and 3.05x improvement in energy-efficiency (defined as GFLOPS/W) for 256, 4096 and 32768 point FFTs, respectively. Compared with the state-of-the-art designs, our implementation realizes up to 7.39x energy efficiency improvement while sustaining almost 20x throughput (defined as MPoints/s) performance.**

I. Introduction

As the FPGA device offers a vast amount of routing, logic and memory resources, various mapping choices can be made when mapping a specific application onto FPGA [1]. This remarkable flexibility creates a huge design space, and results in a big challenge in obtaining the energy optimal design when implementing a given application. Decades of research works have been proposed on digital signal processing application mapping algorithms targeted at FPGA platforms [2], [3], [4]. Those algorithms mainly focus on the optimal mapping solutions to improve performance and resource utilization. Recent research work in [5] presented an application mapping process for FPGA that exploits the embedded memory blocks for computing applications dominated by complex data paths. Energy efficiency is considered as the key metric for performance evaluation. However, the energy efficiency improvement is resulted by reduced programmable interconnect using memories for computing rather than utilizing the power features of embedded FPGA components in the mapping process.

In the application mapping process, design time algorithmic optimizations can be applied step by step bypassing two hierarchical levels: algorithm mapping level and architecture binding level. At the algorithm mapping level, optimizations regardless of implementation details are developed to achieve performance improvement, such as reducing computation or communication complexity, enhancing throughput by parallelization, and saving area by serialization. At the architecture binding level, a considerate solution to bind architecture components to various available hardware resources is critical to optimize energy performance. For mapping a given application how to perform memory binding, pipelining, device activation scheduling from an energy point of view is critical to achieve high energy efficiency.

In this paper, we revisit the classic Fast Fourier Transform (FFT) to identify our proposed algorithmic optimizations for energy efficient designs on FPGAs. Fast Fourier Transform (FFT) is one of the most frequently used image and signal processing kernels, well known for its computational intensity. Over the last decades, a few number of low power and area efficient serialized pipeline FFT architectures have been developed [6], [7], [8], [9], [10]. Those works employ delay feedback or delay commutator design for data permutation between subsequent butterfly computation stages. However, as the data parallelism of a given delay-feedback or commutator architecture is limited by the chosen FFT algorithm, it becomes challenging for those compact serialized designs to meet increasingly demanding throughput requirement nowadays. In this paper, we propose several algorithmic optimizations exploiting the power features of FPGA components to achieve an energy efficient FFT architecture. We apply the optimizations at both the algorithm mapping level and the architecture binding level to demonstrate our design flow. Our architecture is developed with a flexible data parallelism at the expense of extra acceptable latency. The contributions in this paper are summarized as follows:

1) High level energy optimizations at both the algorithm mapping level and the architecture binding level (Section IV).
2) Algorithmic level design approaches exploiting power features of embedded FPGA components (Section IV).
3) A design flow to develop an energy efficient floating-point FFT architecture on FPGA (Section III).
4) Significant energy efficiency improvement compared with the state-of-the-art FFT designs (Section VI).

The rest of the paper is organized as follows. Section II covers the background and related work. Section III introduces the architecture overview and its implementation on FPGA. Section IV describes the high level energy optimizations. Section V details the architecture binding approach. Section VI presents performance comparisons. Section VII concludes the paper.

This work has been funded by DARPA under grant number HR0011-12-2-0023.

II. Background and Related work

Serialized pipeline FFT architectures have been very popular in the last two decades due to its high area/resource efficiency and low I/O (input/output) bandwidth requirement [8], [9], [10], [6], [7]. Radix-x Cooley-Tukey algorithm is one of the most popular algorithms for hardware implementation [6], [7], [11]. Most hardware solutions for Radix-x FFT fall into the following categories: delay feedback or delay commutator architectures, such as Radix-2^2 single-path delay feedback FFT [7] and Radix-4 single-path delay commutator FFT [6]. Various delay feedback or delay commutator implementations are employed to permute data between subsequent butterfly computation stages in FFT. Those compact architectures are more suitable to be implemented as an ASIC accelerator rather than mapped onto FPGA platforms when considering the nontrivial device static power consumption. On the other hand, the achievable maximum throughput using their design is limited due to the customized delay feedback/commutator implementation.

To enhance the design throughput, researchers have proposed several high throughput architectures [12], [13], [14], [15], [16]. In [12], a parameterized soft core generator for high throughput FFT was developed. This generator can automatically produce an optimized design with user inputs for throughput and resource constraints. A flexible and extensive permutation network was employed to meet various throughput requirements. However, energy efficiency is not considered in this work. In [13], the author presented a parameterized energy efficient FFT architecture. Their design was optimized to achieve high energy efficiency by varying the architecture parameters. Some low power design techniques, such as clock gating and power gating, were also employed in their work. In those previous works, either energy efficiency is not considered or only circuit-level power optimizations are employed.

In this paper, we propose energy optimizations at both the algorithm mapping level and architecture binding level introduced in Section I. Those optimizations significantly reduce the energy dissipation of the FFT architecture, which increases with data parallelism. In performance comparison with serialized FFT design from Xilinx [17], our implementation outperforms in energy efficiency even with a 4x data parallelism. To meet various throughput requirement, we develop a data permutation unit which is extensive and owns flexible data parallelism, and we further propose a high level optimization for this unit to improve its memory efficiency. We notice that our architecture is also feasible for VLSI implementation and the techniques employed are applicable for other FFT architectures.

III. Architecture Overview

A. Baseline architecture

In this section, we define the floating-point 1D FFT baseline architecture for performance comparison. To obtain an optimal FFT design, we develop and applied certain static optimizations onto the design components to reduce their energy consumption. We decompose an *N*-point FFT architecture into several basic components including radix block, data path permutation (DAP) unit, and twiddle factor computation (TWC) unit. The design of each architecture component relies on the FFT algorithm in use. Fig. 1 shows Radix-4 based 16-point FFT architecture by concatenating the architecture components. Note that the data parallelism of our architecture is not limited by the FFT algorithm in use by introducing the DAP unit. Implementation details of those components will be introduced next.

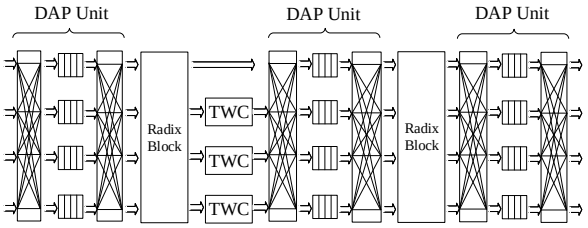


Fig. 1: Architecture Overview for 16-point Radix-4 FFT

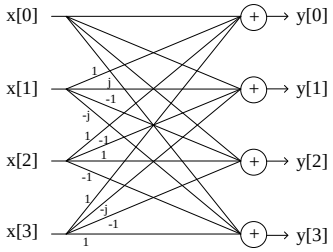


Fig. 2: Raidx-4 block

1) Radix block: The radix block is used to perform a butterfly computation on some input samples. For example, the radix block for radix-4 FFT takes four input samples, performs the butterfly computation and then generates four results in parallel. Each radix block is composed of complex adders and subtractors. The structure of a radix block is determined by the FFT algorithm in use. Fig. 2 shows the structure of radix block for radix-4 FFT.

2) DAP unit: DAP unit is used for data permutation between butterfly computation stages in FFT. This unit enables arbitrary I/O order and processing streaming data. A DAP unit is composed of demultiplexers and data buffers. In subsequent clock cycles, data from previous butterfly computation stage are first multiplexed and written into several data buffers. Each stored data element will be buffered with a certain number of clock cycles and then read out. Outputs from data buffers will also be multiplexed and fed into the next butterfly computation stage. Fig. 3 shows the DAP unit used for a radix-4 based FFT design. Each DAP unit consists of eight 1-to-4 demultiplexers and four data buffers. In each cycle, a data buffer may be read and written simultaneously on different addresses in baseline. Hence each data buffer can be implemented using a dual-port BRAM on FPGA [1]. A data remapping technique devised for DAP unit to reduce memory energy dissipation will be presented in Section IV-B. The size of each data buffer depends on the ordinal number of its present butterfly computation stage and the FFT problem size. Note that each data element is a complex number including both its real part and imaginary part, hence the data width is 64 bit.

3) TWC unit: A TWC unit consists of two parts: the TWC generation logic and the complex number multiplier. As shown in Fig. 4, the TWC generation logic includes several lookup

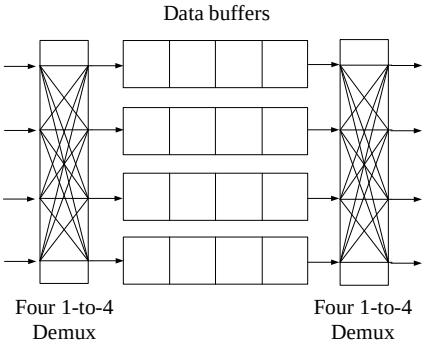


Fig. 3: DAP unit for Radix-4 FFT

tables (ROMs) for storing twiddle factor coefficients, where the data read addresses will be updated with the control signals. The size of each lookup table is determined by the ordinal number of its present butterfly computation stage and the FFT problem size. Each lookup table can be implemented using a BRAM or distributed RAM (dist. RAM) on FPGA [1]. Each complex number multiplier consists of four real number multipliers and two real number adders/subtractors.

B. Architecture binding parameters

At the architecture binding level, we define several parameters to create a design space. In our experiments, two architecture parameters that significantly affect energy efficiency were considered in our design and applied to different architecture components:

- Type of memory element: BRAM versus distributed RAM (dist. RAM) to implement data buffers and twiddle factor lookup tables.
- Pipeline depth: Number of registers to be inserted to increase the pipeline depth for adders/substractors and DSP slices for increased performance versus dynamic power.

According to the FPGA manufacturers user guide [1], BRAM is more power efficient than dist. RAM when used for large size memories. Hence this characteristic can be utilized to trade-off between power and performance for various memory components. As introduced in Section III-A, the size of a data buffer used in DAP unit or a lookup table used in TWC unit varies with the ordinal number of the present butterfly computation stage. Hence we evaluated energy consumption of BRAMs and dist. RAMS with various memory capacities in Section V-A. The corresponding detailed memory binding solutions are introduced in Section V-A.

Traditional pipelining technique can be utilized for power optimization by reducing the supply voltage and maintaining operating frequency. However, the supply voltage is fixed on FPGA boards although two options (0.9 V or 1.0 V) are available statically. But different pipeline depth still has significant impact on power consumption of FPGA-based design due to glitch power and signal power. In our design, floating point arithmetic units are one of the major power consuming components. We divide the floating point arithmetic units into several basic stages, and insert pipeline registers in some of the stages based on the chosen pipeline depth. In this way, we evaluate the affect of the pipeline depth on the power consumption of the floating point arithmetic units. The power evaluation results for floating point units will be discussed in Section V-B.

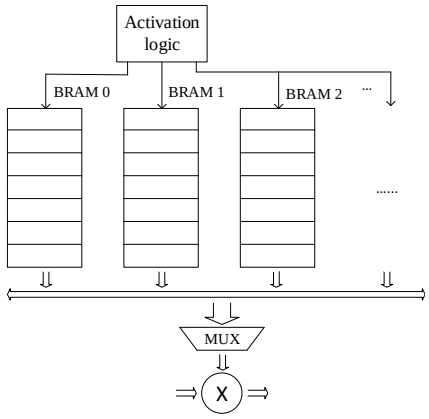


Fig. 4: Memory activation scheduling on TWC units

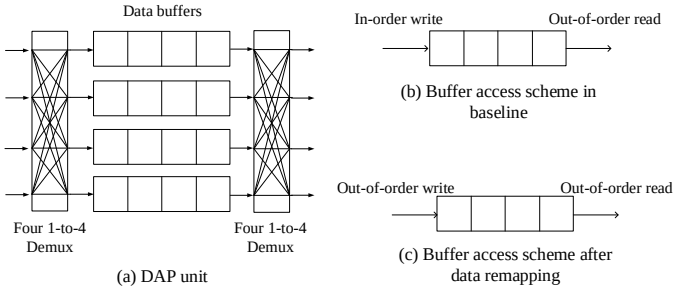


Fig. 5: Buffer access scheme in DAP unit

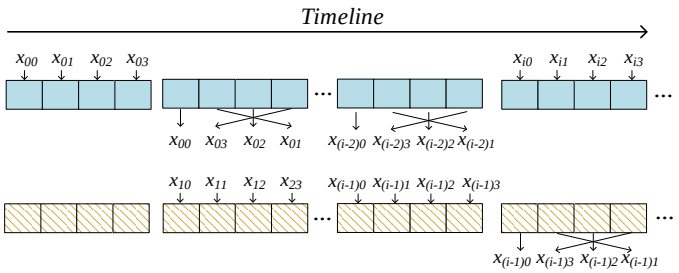
IV. High-level Algorithmic Optimizations

A. Memory activation scheduling

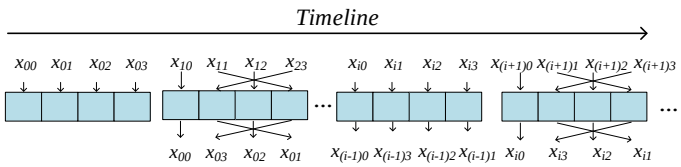
In order to further reduce the energy consumption of memory, the proposed memory activation scheduling technique is introduced as shown in Fig. 4. The memory storing the twiddle factor coefficients is separated into several BRAM blocks, only one of which is activated at every read operation according to the read address. As only one of the BRAM blocks is activated, the energy consumption per read is significantly reduced. This technique can highly improve memory energy performance especially for memories with large capacities in TWC units.

B. Energy efficient data remapping

The DAP unit consists of four data buffers for Radix-4 FFT. Those data buffers are implemented using BRAMs or dist. RAMs to store intermediate data between butterfly computation stages. As shown in Fig. 5 (b), in baseline architecture, each data buffer is written in order (with incremented write addresses) and read out-of-order (with a known sequence of read addresses) simultaneously in subsequent cycles. Correspondingly, a dual-port BRAM is required by each data buffer. To reduce the memory energy consumption, we remap the data when writing the data buffer so that each memory location in the data buffer is written with a new value once the old value is read out. In this way, out-of-order write and read



(a) Buffer access scheme in baseline



(b) Buffer access scheme after data remapping

Fig. 6: Example: data remapping for writing a data buffer in 16-point Radix-4 FFT

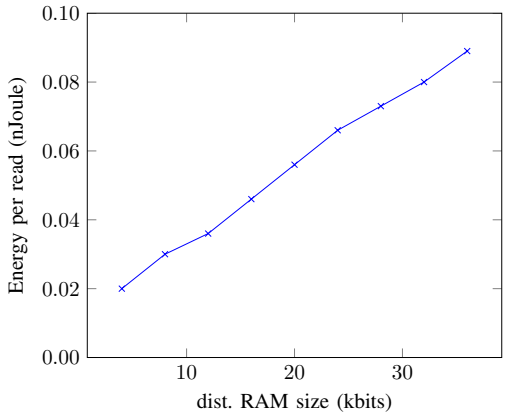


Fig. 7: Energy consumption per read operation for different sizes (dist. RAM) at 200 MHz

operations are performed in parallel in each data buffer. Hence, each data buffer can be implemented using only one single-port BRAM or dist. RAM. As a dual-port memory is actually implemented with two single-port BRAMs or dist. RAMs on FPGA, our proposed data remapping scheme can reduce the memory energy exactly by 50%.

Fig. 6 shows the example of the data buffer access process in baseline and after data remapping for 16-point radix-4 FFT. In Fig. 6(a), two single-port BRAMs are used, which are alternatively read or written. In-order write and out-of-order read are executed. In Fig. 6(b), one single-port BRAM is used, and out-of-order read and write are performed. The BRAM access mode supporting simultaneous read-write operation (read-first) on the same address is employed [1].

V. Architecture Binding Approach

A. Memory binding

In order to optimize the energy consumption of memory, we investigate the energy consumption of dist. RAM and

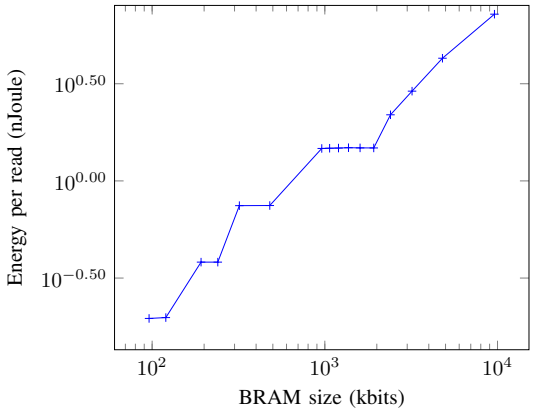


Fig. 8: Energy consumption per read operation for different sizes (BRAM) at 200 MHz

BRAM. We then bind the data buffers in DAP units and the ROMs in TWC units with different memory resources to improve the energy efficiency.

The look-up tables (LUT) in Xilinx FPGA can be implemented as synchronous RAM called dist. RAM. The larger the memory size the more LUTs are required to implement and the routing become more complex, hence the higher energy consumption per read operation. Compare to BRAM, dist. RAM is more energy efficient for implementing small size memory. We investigate the energy consumption of dist. RAM with different size at 200 MHz operating frequency (Fig. 7). The experiment results indicate that the energy per read increases linearly with the dist. RAM size. The BRAM is a dedicated memory resource on FPGAs to implement large size memory. The BRAM on Xilinx 7 series FPGAs stores up to 36 Kbits of data and can be configured as either two independent 18 Kb RAMs, or one 36 Kb RAM [1]. When implementing large memory, many BRAMs are combined together, all of which are activated during read operations. Consequently, the energy consumption of the memory per read is dependent on how many BRAMs are used. We investigate the energy per read for memory with various sizes on a Xilinx FPGA with 32-bit data width operating at 200MHz. As shown in Fig. 8, the energy consumption per read of the memory increases with the size. Based on the results above, we bound all memories with a capacity greater than or equal to 256 to BRAMs, and other memories are bound to dist. RAMs.

B. Pipeline depth

We parameterized the floating point (real number) adder and multiplier by inserting different number of pipeline stages, and at least one arithmetic/logic operation is executed in each pipeline stage. For each floating point adder, its pipeline depth can be varied from 1 to 10. All the floating point adders are implemented with LUTs. For each floating point multiplier, its pipeline depth can be varied from 1 to 7. Each floating point multipliers is implemented using LUTs with two extra DSP slices. The pipeline depth of the multiplier does not include the pipeline stages introduced by the two extra DSP slices and the default pipeline configuration for those IP cores are used. Benefiting from the place and route algorithms for

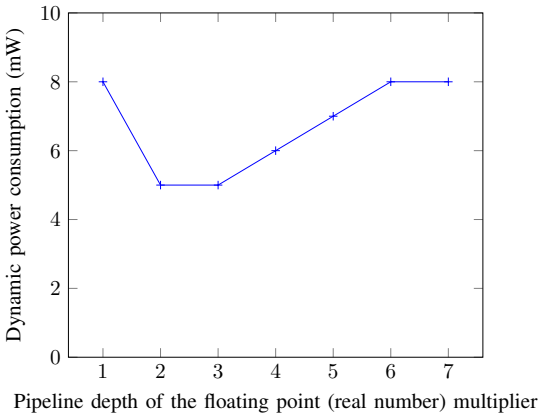


Fig. 9: Dynamic power consumption of the floating point (real number) multiplier with various pipeline depth

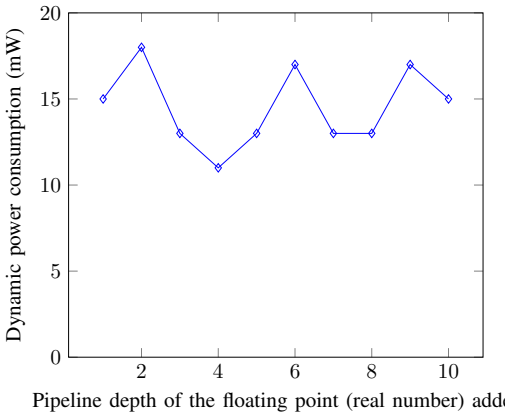


Fig. 10: Dynamic power consumption of the floating point (real number) adder with various pipeline depth

Xilinx FPGAs, all those pipeline depth configurations lead to a maximum operating frequency higher than 200 MHz, which is the minimum operating frequency requirement for our design to achieve a throughput of 800 MPoints/Sec. In the power evaluation experiments, we use 200 MHz as the operating frequency of the floating point units. Fig. 9 and Fig. 10 show the effect of the pipeline depth of the floating point multipliers and adders on energy dissipation respectively. From the figures, the floating point multiplier/adder consumes least amount of dynamic power when the pipeline depth is selected to be three/four. The results above are used for optimizations for both TWC units and radix blocks. The experimental results of the entire FFT architecture will be introduced in Section VI.

VI. Performance comparison

In this section, we compare the energy efficiency of the optimized FFT architecture and the baseline architecture with a sustained throughput of at least 800 MPoints/Sec for the 256, 4096 and 32768 size FFT, respectively. We further conduct performance comparison with the state-of-the art designs. The experiments were conducted on the state-of-the-art Xilinx Artix 7 XC7A200TL with -2L speed grade [1]. The experiments were performed using Xilinx ISE 14.4 development tools [18] , including the Xilinx ISE XPower Analyzer tool for determining the power dissipation. The designs were verified by post place-and-route simulation. We created input vectors which result in an average switching activity of 12.5% for floating point input samples. All the reported results are based on post place-and-route simulations. We used the VCD (Value Change Dump Format) file as input to the Xilinx ISE XPower Analyzer to produce accurate power dissipation estimations. Clock power and I/O power are also considered when calculating the dynamic power dissipation. We assume that the FFT implementation will be used as part of the pre-processing component of as SOC system, such as Xilinx Zynq-7000 [1]. The output will be directly fed into other on-chip components, hence only input ports power is included in I/O power. As energy efficiency is not affected by the frequency when supply voltage is a constant, we fixed operating frequency at 200 MHz in experiments.

A. Energy performance of the complete optimized FFT architecture

TABLE I: Energy performance (dynamic power) of the optimized FFT architecture

	Baseline architecture			Optimized architecture			
FFT size	Power (W)	Energy per point (nJoule)	Energy efficiency (GFLOPS/W)	Power (W)	Energy per point (nJoule)	Energy efficiency (GFLOPS/W)	Energy efficiency improvement
256	4.03	5.03	5.86	1.41	1.69	17.45	2.97x
4096	6.85	8.56	5.43	2.37	2.86	16.24	2.99x
32768	10.94	13.68	4.64	3.65	4.47	14.19	3.05x

1) Comparison with baseline architecture: The energy efficiency of the baseline and the optimized architecture is shown in Table I. The optimized architecture achieves 2.97x, 2.99x and 3.05x energy efficiency improvement for 256, 4096 and 32768 point FFT, respectively. In this comparison, we included dynamic power components consisting of logic power, routing power, memory power, clock power and I/O power. In baseline architecture, the memory binding is automatically done by the Xilinx ISE tool and the floating point units are fully pipelined. None of the proposed optimizations were applied in baseline architecture. We employed 200 MHz as the operating frequency for both designs when conducting power evaluation.

TABLE II: Energy performance (dynamic power) comparison with Xilinx FFT IP Cores

	Xilinx FFT IP Core 7.1		Optimized architecture			
FFT size	Throughput (MPoints/s)	Energy efficiency (GFLOPS/W)	Throughput (MPoints/s)	Energy efficiency (GFLOPS/W)	Throughput improvement	Energy efficiency improvement
256	45.8	8.68	800	17.45	17.5x	2.01x
4096	44.1	8.11	800	16.24	18.1x	2.00x
32768	39.9	1.92	800	14.19	20.1x	7.39x

2) Comparison with Xilinx FFT IP Cores: We also benchmark our performance against Xilinx FFT IP Core 7.1 available in Xilinx ISE [17]. The Xilinx FFT IP Core is configured

so that 3-multiplier structure is used for implementing each complex multiplier, and CLB logic is used for implementing butterfly arithmetic. Radix-4 solution with burst I/O is employed. Both the data input and output are in natural order. The input data samples are in IEEE-754 single-precision (32-bit) floating-point format. Note that only single channel mode is supported by Xilinx floating-point FFT IP Core. Our implementation outperformed Xilinx FFT IP cores for all the selected problem sizes in terms of energy efficiency when considering dynamic power. The energy efficiency of the Xilinx FFT IP Cores and the optimized architecture are calculated and listed in Table II. The optimized architecture achieves 2.01x, 2.00x and 7.39x energy efficiency improvement for 256, 4096 and 32768 point FFT, respectively. In order to give a more thorough view of the energy efficiency of the FFT architecture, we further evaluate the static power dissipation of the architecture and perform the performance comparison when including the static power dissipation.

TABLE III: Performance comparison with SPIRAL IP Cores

	Baseline architecture			Optimized architecture			
FFT size	Power (W)	Energy per point (nJoule)	Energy efficiency (GFLOPS/W)	Power (W)	Energy per point (nJoule)	Energy efficiency (GFLOPS/W)	Energy efficiency improvement
256	2.84	3.55	8.29	1.41	1.69	17.45	2.10x
4096	4.67	5.84	7.96	2.37	2.86	16.24	2.04x
32768	9.25	11.56	5.49	3.65	4.47	14.19	2.58x

3) Comparison with SPIRAL FFT IP Cores: We also employ SPIRAL FFT IP Cores from Carnegie Melon University [12] to compare with our designs. The SPIRAL FFT IP Cores are throughput highly optimized FFT architectures. Their FFT soft IP Cores are automatically generated in synthesizable RTL Verilog code with user inputs. In our experiments, we fixed the number of channels (each channel represents an complex number input) to be four for both designs. The energy efficiency of the SPIRAL FFT IP Core and the optimized architecture are calculated and listed in Table III. The experimental results show that our optimized architecture achieves 2.10x, 2.04x and 2.58x energy efficiency improvement for 256, 4096 and 32768 point FFTs, respectively.

VII. Conclusion

In this paper, we systematically proposed several algorithmic optimizations for an energy efficient throughput-oriented FFT architecture on FPGA. By developing high level energy optimizations and applying them onto specific components in FFT designs, we significantly reduce the dynamic power consumption, which increases with data parallelism. The experimental results comparing with the advanced commercial IP Cores show that our implementation outperforms in energy efficiency even with a 20x throughput. Note that although only the optimized FFT architecture is presented, the proposed algorithmic optimizations are also applicable for other signal processing applications. In the future, we plan to build a design framework targeted at several signal processing kernels, which enables automatic algorithmic optimizations addressing technology scaling.

References

[1] "XST user guide for Virtex-6, Spartan-6, and 7 series devices," http://www.xilinx.com/support/documentation.

[2] G. D. Micheli, *Synthesis and optimization of digital circuits*. McGraw-Hill Higher Education, 1994.

[3] D. Zaretsky, G. Mittal, X. Tang, and P. Banerjee, "Overview of the FREEDOM compiler for mapping DSP software to FPGAs," in *Proc. of IEEE International Conference on Field-Programmable Custom Computing Machines (FCCM)*, April 2004, pp. 37–46.

[4] S. Mohanty and V. K. Prasanna, "A Model-based Extensible Framework for Efficient Application Design Using FPGA," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 2, Apr. 2007.

[5] A. Ghosh, S. Paul, and S. Bhunia, "Energy-Efficient Application Mapping in FPGA through Computation in Embedded Memory Blocks," in *Proc. of IEEE International Conference on VLSI Design (VLSID)*, Jan 2012, pp. 424–429.

[6] G. Bi and E. Jones, "A pipelined FFT processor for word-sequential data," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp. 1982–1985, 1989.

[7] S. He and M. Torkelson, "A new approach to pipeline FFT processor," in *Proc. of IPPS '96*, pp. 766–770.

[8] A. Cortes, I. Velez, and J. Sevillano, "Radix r^k FFTs: Matricial Representation and SDC/SDF Pipeline Implementation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2824–2839, July 2009.

[9] T. Cho, H. Lee, J. Park, and C. Park, "A high-speed low-complexity modified radix-25 FFT processor for gigabit WPAN applications," in *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2011, pp. 1259–1262.

[10] M. Garrido, K. Parhi, and J. Grajal, "A Pipelined FFT Architecture for Real-Valued Signals," *IEEE Transactions on Circuits and Systems*, vol. 56, no. 12, pp. 2634–2643, Dec 2009.

[11] L. R. Rabiner and B. Gold, "Theory and application of digital signal processing," *Englewood Cliffs, NJ, Prentice-Hall, Inc.*, vol. 1, p. 777, 1975.

[12] G. Nordin, P. Milder, J. Hoe, and M. Puschel, "Automatic generation of customized discrete fourier transform IPs," in *Proc. of DAC*, 2005, pp. 471–474.

[13] S. Choi, G. Govindu, J.-W. Jang, and V. K. Prasanna, "Energy-efficient and parameterized designs for fast Fourier transform on FPGAs," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing 2003 (ICASSP'03)*, vol. 2, pp. II–521.

[14] S.-N. Tang, J.-W. Tsai, and T.-Y. Chang, "A 2.4-GS/s FFT Processor for OFDM-Based WPAN Applications," *IEEE Transactions on Circuits and Systems*, vol. 57, no. 6, pp. 451–455, June 2010.

[15] C. Cheng and K. Parhi, "High-Throughput VLSI Architecture for FFT Computation," *IEEE Transactions on Circuits and Systems*, vol. 54, no. 10, pp. 863–867, Oct 2007.

[16] L. Liu, J. Ren, X. Wang, and F. Ye, "Design of Low-Power, 1GS/s Throughput FFT Processor for MIMO-OFDM UWB Communication System," in *Proc. of IEEE International Symposium on Circuits and Systems*, May 2007, pp. 2594–2597.

[17] "Xilinx LogiCORE IP Fast Fourier Transform v7.1," http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf.

[18] "Vivado Design Suite User Guide: Design Flows Overview," http://www.xilinx.com/support/documentation/sw_manuals/xilinx2013_4/ug892-vivado-design-flows-overview.pdf.