High Throughput Energy Efficient Parallel FFT Architecture on FPGAs

Ren Chen
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, USA 90089
Email: renchen@usc.edu

Neungsoo Park
Department of Computer Science and Engineering
Konkuk University
Seoul, South Korea, 143-701
Email: neungsoo@konkuk.ac.kr

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

***Abstract*—Throughput is a key performance metric for streaming FFT architectures. However, increasing spatial parallelism to improve throughput introduces complex routing, thus resulting in high power consumption. In this paper, we propose a high throughput energy efficient parallel FFT architecture based on Cooley-Tukey algorithm. Multiple pipeline FFT processors using time-multiplexing are utilized to perform FFT computation tasks in parallel. This design realizes high performance using task-level parallelism and avoids complex routing. Furthermore, to reduce the memory power consumption, a periodic memory activation (PMA) scheme is developed. By analyzing energy efficiency (defined as GOPS/Joule) asymptotically, we show that our design achieves a low energy efficiency complexity while satisfying a high-throughput requirement. For N-point FFT ($64 \leq N \leq 4096$), our proposed architecture achieves $50 \sim 63$ GOPS/Joule, i.e., up to 78% of the *Peak Energy Efficiency* of FFT designs on FPGAs. Compared with a state-of-the-art design, our design improves the energy efficiency (defined as GOPS/Joule) by 17% to 26% with the same throughput.**

I. Introduction

FPGA is a promising implementation technology for computationally intensive applications such as signal, image, and network processing tasks [1], [2], [3]. Fast Fourier Transform (FFT) is one of the frequently used kernels in a variety of image and signal processing applications. High throughput FFT modules are widely used for modulation and demodulation in such applications [4]. To meet throughput requirements, maximizing performance under power dissipation constraints is a critical issue. The power consumption of an FFT implementation depends on the number of memory modules, processing units and the interconnection topology.

By maximizing the degree of spatial parallelism, fully spatial FFT architecture based on the Cooley-Tukey algorithm can boost throughput [5]. However, it achieves high performance at the expense of much routing resources. As FFT problem size grows, unfolding the architecture spatially becomes infeasible due to considerable routing power and resource consumptions caused by intricate interconnections.

Pipeline FFT processors are a class of compact FFT hardware architectures which compute the FFT sequentially using time-multiplexing [6], [7], [8]. Time-multiplexing folds fully spatial parallel FFT architecture vertically, thus avoiding complex interconnections. A pipeline FFT processor is composed of sequentially concatenated butterfly units and data buffers. This processor supports non-stop processing with data fed in a streaming manner. During the processing of one FFT task, each butterfly unit is reused several times by continually incoming data streams. High computational performance per unit area can be achieved by these designs, while the throughput is limited owing to partial spatial parallelism.

In this paper, we propose FFT architectures based on hybrid designs using time-multiplexing and spatial parallelism. Multiple pipeline FFT processors are utilized to meet a throughput requirement by increasing task parallelism. Routing power is reduced by time-multiplexing at the expense of extra data buffers. For each processor, partially spatial parallelism is employed to enhance the throughput. A periodic memory activation (PMA) scheme is developed to reduce the memory power dissipation. We also perform an analysis on the energy efficiency asymptotically, and evaluate the potential *Peak Energy Efficiency* of FFT designs on FPGAs. The experimental results show that our architecture achieves a high energy efficiency by using the proposed design approaches. The contributions in this paper are summarized as follows:

1) A high throughput energy efficient FFT architecture using time-multiplexing and spatial parallelism (Section III-A).
2) A PMA scheme that significantly reduces the memory power dissipation in the proposed architecture (Section III-C).
3) An asymptotic analysis on energy efficiency for demonstrating the improved performance compared with the fully spatial parallel architecture (Section IV).
4) Implementations which sustain up to 78% of the upper bound on energy efficiency (*Peak Energy Efficiency*) of FFT designs on the target platform (Section V-B).
5) Significant energy efficiency improvement compared with a state-of-the-art FFT design (Section V-E).

The rest of the paper is organized as follows. Section II covers the background and related work. Section III introduces the proposed architecture and its implementation on FPGA. Section IV describes the asymptotic analysis on energy effi-

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

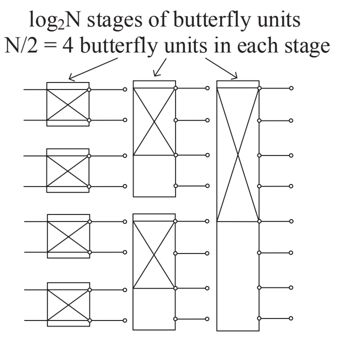

$\log_2$N stages of butterfly units
N/2 = 4 butterfly units in each stage

Fig. 1: 8-point fully spatial parallel FFT

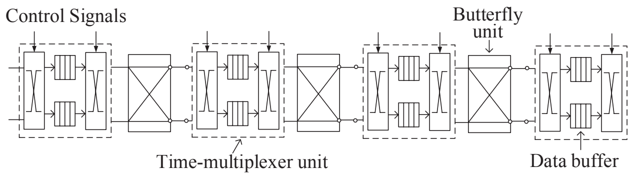


Fig. 2: Pipeline FFT processor for Radix-2 based 8-point FFT

ciency. Section V presents experimental results and analysis. Section VI concludes the paper.

II. Background and Related Work

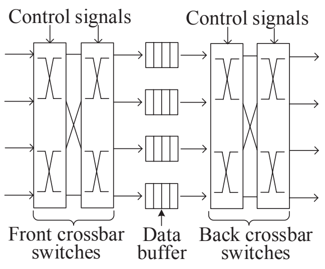
A. Background

Given N complex numbers $x_0, ..., x_{N-1}$, Discrete Fourier transform (DFT) is computed as:

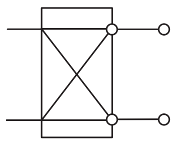
$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}},\ k = 0, ..., N-1. \quad (1)$$

Radix-x Cooley-Tukey FFT is a well-known decomposition based algorithm for DFT [3]. Fully spatial parallel architecture is a direct mapping of Cooley-Tukey algorithm. Fig. 1 shows the 8-point fully spatial parallel FFT architecture using Radix-2 FFT. Three stages of concatenated butterfly units are used. Data are permuted between each stage. This architecture can be pipelined to take eight inputs and produce eight outputs in every cycle. For N-point FFT, $\log_2 N$ stages of butterfly units are required. As N grows, due to increasing wire length between stages for permutation, large interconnection power consumption becomes a serious design issue [9].

Time-multiplexing can be employed to fold fully spatial parallel FFT architecture vertically, thus saving resource and area overhead. Using time-multiplexing, we can design a pipeline FFT processor flexibly to meet diverse throughput requirements and resource constraints. Fig. 2 shows a pipeline FFT processor with a throughput of two points per cycle for Radix-2 based 8-point FFT. A time-multiplexer unit is used in each stage for data permutation so that data are fed through the butterfly unit in the correct order. Each time-multiplexer unit is composed of data buffers and crossbar switches. It enables each butterfly unit to be reused by continually incoming data streams. In this paper, by expanding this design, multiple pipeline FFT processors are designed to perform high throughput FFT computations without intricate interconnections.



(a) Time-multiplexer unit



(b) Butterfly unit

Fig. 3: Architecture building blocks

B. Related Work

An energy efficient 1024-point FFT processor was developed in [6]. Cache-based FFT algorithm was proposed to optimize the power of a FFT processor, which supports two operation modes including high performance mode and low power mode. In [7], a high-speed and low-power FFT architecture was presented. A delay balanced pipeline FFT architecture was proposed based on split-Radix algorithm. Algorithms for reducing computation complexity were explored. However, these works mainly focuses on maximizing design operating frequency rather than exploiting parallelism to obtain high performance. Based on Radix-x FFT, various pipeline FFT architectures have been proposed, such as Radix-2 single-path delay feedback FFT [8], Radix-4 single-path delay commutator FFT [3], Radix-2 multi-path delay commutator FFT [1], and Radix-2^2 single-path delay feedback FFT [10]. These architectures achieved superior performance per unit area with various spatial parallelism, while throughput was not considered.

A parameterized soft core generator for high throughput DFT was developed in [2]. The generator can automatically produce an optimized design with user inputs for performance and resource constraints. However, energy efficiency was not discussed. In [11], a parameterized FFT architecture was presented. To identify energy-efficient designs, performance estimation and design trade-offs were performed. Power was optimized by using design techniques such as clock gating but throughput was not considered. In this paper, we propose a high throughput energy efficient parallel FFT architecture. An asymptotic analysis on energy efficiency is performed to demonstrate the improved performance. Design approaches such as time multiplexing and the PMA scheme are developed to reduce the routing power and the memory power respectively, thus improving energy efficiency.

III. Architecture and Optimizations

A. Architecture Building Blocks

Multiple pipeline FFT processors are utilized in our design. Each processor is implemented based on Radix-x Cooley-Tukey FFT algorithm. It consists of two types of building blocks (see Fig. 3): Butterfly unit and time-multiplexer unit. A complete design for a given throughput requirement can be obtained using a combination of the basic blocks.

1) Time-multiplexer Unit: This unit is composed of crossbar switches and data buffers. Fig. 3(a) shows the time-multiplexer unit for Radix-4 based 16-point FFT. Four data inputs are firstly permuted by the front crossbar switches, and then temporarily stored in the four data buffers. After sixteen

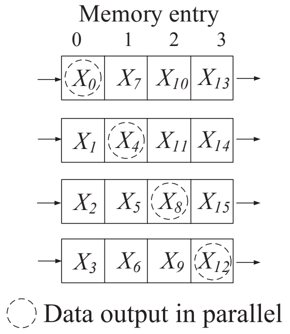

Memory entry
0 1 2 3
X_0 X_7 X_{10} X_{13}
X_1 X_4 X_{11} X_{14}
X_2 X_5 X_8 X_{15}
X_3 X_6 X_9 X_{12}
Data output in parallel

Fig. 4: Data layout in the data buffers for Radix-4 based 16-point FFT

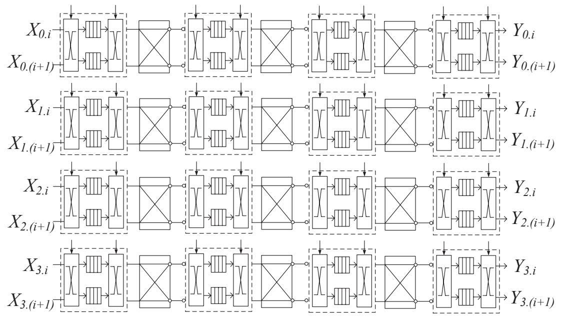


Fig. 5: High throughput 8-point parallel FFT architecture ($V_p = 2, N_p = 4, Th = 8$)

data elements enter the data buffers, they are then permuted by the back crossbar switches and read out over four cycles.

Fig. 4 shows the data layout in the data buffers for Radix-4 based 16-point FFT architecture. Data inputs $(X_0, X_1, .., X_{15})$ enter into the data buffers sequentially over four cycles. In clock cycle i $(0 \leq i \leq 3)$, four data inputs $(X_{4i+0}, X_{4i+1}, X_{4i+2}, X_{4i+3})$ are permuted by the front crossbar switches and then fed into the ith entries of each data buffer. After that, data at a stride of four $(X_i, X_{i+4}, X_{i+8}, X_{(i+12)}, i = 0, 1, 2, 3)$ are stored in different data buffers and then output in parallel. For example, X_0, X_4, X_8, and X_{12} are read out simultaneously. The number of data buffers required for each time-multiplexer unit is determined by the degree of spatial parallelism, i.e., the number of parallel inputs of each processor. Assuming the number of data buffers used in each stage is k, then the size of each data buffer is N/k.

2) Butterfly Unit: This unit executes butterfly computations and twiddle factor multiplications. Twiddle factor coefficients are stored in block RAMs (BRAMs) on FPGAs. Fig. 3(b) shows the butterfly unit for Radix-2 FFT, where the inside circles represent complex number adders and the outside circles are complex number multipliers. For Radix-2 FFT, each butterfly unit needs six real adder/subtractors and four real multipliers to compute butterfly operations. It takes two inputs and generates two outputs in parallel.

B. High throughput FFT Architecture

In our design, multiple processors are used and each is responsible for an N-point FFT task. To implement the proposed architecture, three mapping parameters are considered:

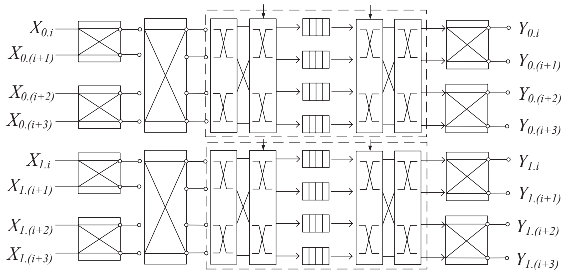


Fig. 6: High throughput 8-point parallel FFT architecture ($V_p = 4, N_p = 2, Th = 8$)

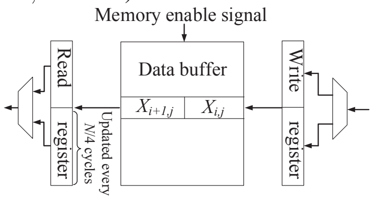


Fig. 7: Block diagram of the data buffer using the PMA scheme

- *Vertical parallelism (V_p)*: determines the number of parallel inputs of each pipeline FFT processor. It also represents the degree of spatial parallelism of one processor.
- *Number of FFT processors (N_p)*: determines the number of pipeline FFT processors used in the design. As each processor is responsible for one FFT computation task, N_p also represents the task parallelism.
- *Throughput (Th)*: determines the number of parallel data inputs in each clock cycle. $Th = V_p \times N_p$.

The parameters above are chosen to create a specified design. Two design examples for 8-point FFT are presented in Fig. 5 and Fig. 6. In Fig. 5, $V_p = 2$, $N_p = 4$, $Th = 8$, four pipeline processors are used. This architecture achieves high throughput benefiting from task parallelism. In Fig. 6, $V_p = 4$, $N_p = 2$, $Th = 8$, two pipeline processors are employed. The higher spatial parallelism increases throughput and requires less number of data buffers. These two architectures achieve the same throughput while they can satisfy different routing and memory resource constraints.

C. PMA scheme

Data buffer is one of the main components consuming considerable power in our design. As data buffers are implemented in dual-ported BRAMs on FPGAs, reducing BRAM power makes a huge impact on the power dissipation of the entire design. A BRAM block can be disabled/deactivated to suppress the switch activity, thus reducing the BRAM power [12]. Hence, we propose a PMA scheme to deactivate the BRAM periodically when it is not accessed. Supposes the size of a data buffer is n, and data width is one byte. Then we can reorganize it as a memory of size (n/m) and data width is m-byte. Fig. 7 shows the block diagram of the data buffer when $m = 2$. Here we select the value of m empirically. Two extra two-byte registers are used for the read/write BRAM operation respectively. Read and write operations are performed simultaneously in each dual-ported BRAM. As shown in the figure, the write register is configured as serial-in-parallel-out shift register. In each cycle, the write register shifts the incoming one-byte data into its parallel register. In every two cycles, the two-byte data are written into the BRAM in parallel. Hence, the BRAM can be deactivated once in every two cycles during a write operation. Note that data access pattern for writing BRAM is regular. However, irregular access is required when reading BRAM. The read register is configured as parallel-in-serial-out shift register. In every two cycles, two-byte data fetched from the BRAM are written into the read register. The least significant byte is shifted out in each cycle. The most significant byte is updated every $N/4$ (BRAM access stride) cycles. For this purpose, a counter is required. When this byte is consumed and then updated, the BRAM can be deactivated. Hence the BRAM can be deactivated once in every $N/4$ cycles during a read operation.

IV. Energy Efficiency Analysis

In this section, we perform an analysis on energy efficiency asymptotically. Energy efficiency is compared between our design and the fully spatial parallel architecture. For the fully spatial parallel architecture, the throughput is N. To achieve the same throughput, we employ $N/4$ pipeline FFT processors ($V_p = 4$) for various N. We use a set of notations and symbols as follows:

TABLE I: Notations and Symbols

Description	Symbol
FFT problem size	N
Number of real operations required to compute an N-point FFT	$R(N)$
Power consumption of an FFT design	$P(N)$
Average time used for one FFT computation	$T(N)$
Energy efficiency of an FFT design	$E(N)$
Average number of activated computation units (arithmetic unit or DSP slice) per clock cycle	$n_c(N)$
Average power consumption of each computation unit	p_c
Computation power in an FFT design	$P_c(N)$
The total wire length used between butterfly stages of an FFT design	$l_w(N)$
Average power consumption of unit routing wire	p_w
Interconnection power in an FFT design	$P_w(N)$
Average number of activated memory blocks per clock cycle	$n_m(N)$
Average power consumption of each memory block	p_m
Memory power in an FFT design	$P_m(N)$

To calculate the entire power, three major types of power are considered: *computation power* ($P_c(N)$), *interconnection power* ($P_w(N)$), and *memory power* ($P_m(N)$). Note that the I/O power is not considered as we assume original data are on-chip. Hence $P(N)$ is computed as

$$P(N) = P_c(N) + P_w(N) + P_m(N) \tag{2}$$

An N-point FFT requires $O(N \log N)$ real operations, i.e., $R(N)$ is $O(N \log N)$ [5]. Then *Energy Efficiency* $E(N)$ can be calculated as:

$$E(N) = \frac{Number\ of\ operations}{Time \times Average\ power} = \frac{R(N)}{T(N) \times P(N)} \tag{3}$$

A. Computation power

When given an certain operating frequency and IP Cores for basic arithmetic operations, p_c is almost a constant with a specified signal toggle rate. We can estimate:

$$P_c(N) = n_c(N) \times p_c \tag{4}$$

As $Th = N$ for the two target architectures, $O(N \log N)$ butterfly units are required. Hence $n_c(N)$ needs to be $O(N \log N)$ and the *computation power* $P_c(N)$ can be computed as $O(N \log N)$ for both of them.

B. Interconnection power

Routing power on FPGAs almost increases linearly with wire length [13]. Hence we can estimate:

$$P_w(N) = l_w(N) \times p_w \tag{5}$$

For Radix-2 based fully spatial parallel architecture, there exits $\log_2 N$ stages of interconnections, and the wire length is $2^n (1 \leq n \leq \log_2 N/2)$ in each stage. Hence its total wire length $l_w(N)$ can be computed as $\alpha N[2 + 4 + ... + N/2] = O(N^2)$, α is a constant [14]. In our design, $O(N)$ processors are required, and the wire length between stages is constant for each processor. Thus $l_w(N)$ is $O(N \log N)$. Based on Eq. (5), $P_w(N)$ can be computed as $O(N^2)$ and $O(N \log N)$ respectively for the fully spatial parallel architecture and our proposed architecture.

C. Memory power

Two types of memories, including distributed RAM (dist. RAM) and BRAM, exist on FPGA. As BRAM is highly optimized in area, it consumes less power per bit than dist. RAM. Data buffers are implemented in BRAMs. According to [12], despite how small the memory size is, a BRAM block has to be assigned. Therefore, BRAM power is determined by the number of BRAM blocks used rather than the total size of memory [12]. Thus we obtain:

$$P_m(N) = n_m(N) \times p_m \tag{6}$$

where p_m is determined by the factors such as operating frequency, access enable rate, and access data width [12]. Decreasing access enable rate can reduce p_m. The PMA scheme presented in Section III-C is proposed based on this observation. In our design, each of the $O(N)$ processors needs $O(\log N)$ data buffers. Therefore, the total number of data buffers implemented in BRAMs (n_m) is $O(N \log N)$. For the fully spatial parallel architecture, BRAMs storing twiddle factor coefficients are used in each stage, hence N_m is also $O(N \log N)$. Therefore, memory power $P_m(N)$ can be computed as $O(N \log N)$ for both architectures.

D. Energy efficiency

Based on the analysis above, the energy efficiency is calculated using Eq. (3). $R(N)$ has been computed as $O(N \log N)$. As $Th = N$, T is a constant for both architectures. Table II shows the complexity of energy efficiency. The complexity of energy efficiency of the fully spatial parallel architecture is $O(\log N/N)$, which indicates that the energy efficiency will dramatically decreases as problem size grows. For our architecture, the complexity of energy efficiency is $O(1)$. Thus the energy efficiency of our design decreases much slower than the fully spatial parallel architecture. This indicates that when given a throughput target increasing with N, our design can be optimized to sustain a high energy efficiency.

V. Experimental Results and Analysis

In this section, we compare the performance of the proposed architecture with both a baseline architecture and a state-of-the-art design. *Energy Efficiency* (defined in Eq. (3)) and

TABLE II: Complexity of energy efficiency

	$P_c(N)$	$P_w(N)$	$P_m(N)$	$E(N)$
Fully spatial parallel architecture	$O(N \log N)$	$O(N^2)$	$O(N \log N)$	$O(\log N/N)$
Our proposed architecture	$O(N \log N)$	$O(N \log N)$	$O(N \log N)$	$O(1)$

Throughput (defined in Section III-B) are considered as two performance metrics in the experiments.

A. Experimental Setup

Radix-4 FFT is used for implementations. The designs are implemented in Verilog on Virtex-7 FPGA (XC7VX980T, speed grade -2L) using Xilinx ISE 14.4. Xilinx LogiCORE IP Cores v11.2 are used for implementation of adders or multipliers. The input test vectors (16-bit fixed-point) for simulation were randomly generated and had an average toggle rate of 50%. We used VCD files (value change dump file) as inputs to Xilinx XPower Analyzer to produce accurate power dissipation estimation [12]. We set the operating frequency to 333 MHz for power evaluation.

B. Peak Energy Efficiency

The energy efficiency of any FFT design is upper bounded by the inherent peak performance of the platform. This depends on the target device and the IP cores used to perform the arithmetic operations. We measure this *Peak Energy Efficiency* by using a minimal architecture to depict the ideal conditions. Thus, we ignore all overheads such as memory, I/O, routing, and buffers that may be employed by an implementation. This minimal architecture consists of only multipliers and adders as the basic computation units. Each computation unit has its own input registers for pipelining. Assuming the throughput target is N using Radix-4 FFT, this minimal architecture requires $\frac{11}{4}N\log_2 N$ real adders and $\frac{3}{2}N\log_2 N$ real multipliers respectively [5]. Using the proposed experimental setup, we estimated the power consumption of the adder and the multiplier from Xilinx LogiCORE IP Cores [12]. The power consumption results of 16-bit multiplier and 16-bit adder are 5.28 mW and 3.48 mW respectively. Based on Eq. (3), the *Peak Energy Efficiency* can be computed as 81 GOPS/Joule for Radix-4 based N-point FFT. Note that *Peak Energy Efficiency* is independent on N and Th, and as the multiplier and adder cores improve, we can expect a corresponding increase in the *Peak Energy Efficiency*. We use the *Peak Energy Efficiency* as an upper bound on the performance of the chosen FFT algorithm. We also compare the sustained performance of our designs against this bound in Section V-C.

C. Energy efficiency and power evaluation

We conducted the experiments for our proposed architecture by varying N, and compare the performance results with a baseline architecture. N varies from 64 to 4096. Both the two architectures achieve the same throughput Th. As available resources are limited on the target device, Th is empirically chosen as $2^{\log_4 N}$. V_p is selected to be 4 in our design with a minimal degree of spatial parallelism. The baseline architecture is a pipeline FFT processor, which has a degree of spatial parallelism maximized to Th.

The energy efficiency results of the two target architectures are presented in Fig. 8. When $N = 64$, the baseline architecture achieves higher energy efficiency than our proposed

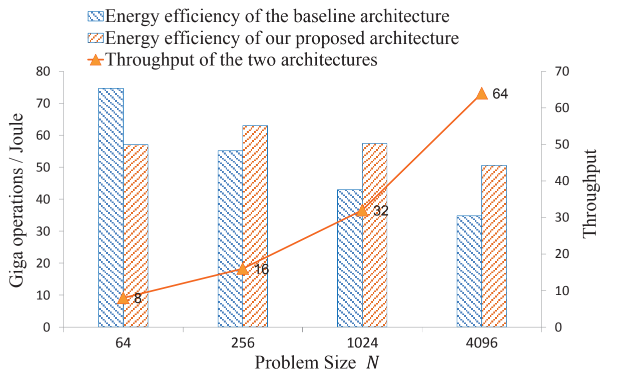


Fig. 8: Energy efficiency of our proposed architecture ($V_p = 4, N_p = Th/4$) and the baseline architecture

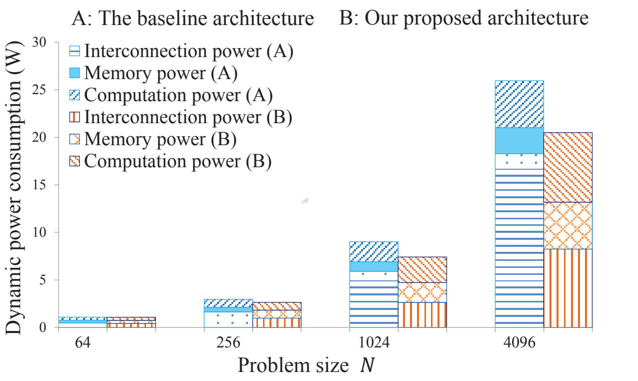


Fig. 9: Power profile of our proposed architecture ($V_p = 4, N_p = Th/4$) and the baseline architecture

architecture. The reason for that is interconnection power is not dominant in the total power consumption when N is small. However, as N increases, the energy efficiency of the baseline architecture significantly decreases, which matches with our energy efficiency analysis in Section IV-D. The experimental results also show that our proposed architecture can achieve $50 \sim 63$ GOPS/Joule for various N. We also observe a performance turning point ($N = 256$) resulted from increasing memory and interconnection power consumption.

Fig. 9 shows the power profile of the two target architectures. For the baseline architecture, the interconnection power grows rapidly when the throughput increases. As N grows progressively from 64 to 4096, the interconnection power of the baseline increases from 44% to 70% of its total dynamic power consumption, and increases from 1.5x to 2.5x interconnection power consumed by our proposed design. This result is consistent with the analysis of interconnection power in Section IV-B. For our design, the interconnection power accounts for $33\% \sim 38\%$ of the total power for various N, i.e., only increases within a narrow range by percentage. p

D. Evaluation of PMA scheme

To evaluate the proposed PMA scheme, the read/write buffer size is empirically chosen to be two. Problem size N varies from 64 to 4096. V_p is selected to be 4. Fig. 10 shows the total dynamic power consumption of one pipeline processor in our proposed architecture using and without using the PMA scheme. It shows that this scheme can reduce the total dynamic

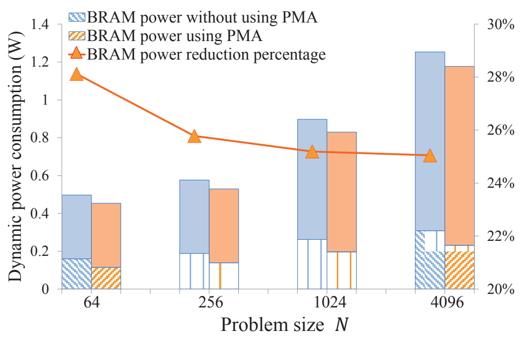


Fig. 10: Dynamic power consumption of a pipeline FFT processor using and without using the proposed PMA scheme

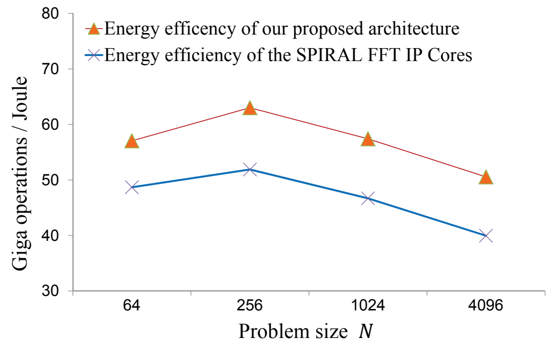


Fig. 11: Energy efficiency of the proposed architecture ($V_p = 4$, $N_p = 8$, $Th = 32$) and the SPIRAL FFT IP Cores

power consumption of each FFT processor by up to 10%. It also shows that the BRAM power can be reduced by 25% to 28%. We observe that as N grows, the BRAM power reduction percentage decreases. This indicates that the PMA scheme is sensitive to N, since the BRAM block cannot be deactivated most of the time during the read operation for large N. Note that the proposed PMA scheme is scalable if more processors are required to achieve a higher throughput.

E. Performance Comparison

We finally use SPIRAL FFT IP cores for performance comparison with our proposed architecture [2]. The SPIRAL FFT IP cores are soft IP cores based on streaming FFT architectures. The interconnection network in their design has been mathematically proved to be control-cost optimal [15]. They developed a tool to automatically generate customized FFT soft IP cores in synthesizable RTL Verilog with user inputs, including transform size, data ordering, data precision, and streaming width, i.e., throughput Th [2]. In this experiment, Th is set to 32 considering the available resources on the target device. Input vectors are 16-bit fixed point data presented in their natural ordering. Transform size varies from 64 to 4096. We use $V_p = 4$ and $N_p = Th/4$ in our design to achieve the same throughput of SPIRAL FFT IP Cores for various transform size. As shown in Fig. 11, for various N ($64 \leq N \leq 4096$), our proposed architecture achieves $50 \sim 63$ GOPS/Joule, and improves energy-efficiency by 17% to 26%, compared with the SPIRAL FFT IP Cores. Compared to the upper bound on energy efficiency of FFT (discussed in Section V-B), our architecture achieves up to 78% of the *Peak Energy Efficiency* (81 GOPS/Joule). This result indicates that our design sustain high percentage of the upper bound on energy efficiency of FFT.

VI. Conclusion

In this work, we presented a high-throughput energy efficient architecture using Radix-x Cooley-Tukey FFT algorithm. The proposed architecture achieves high throughput by increasing both task and spatial parallelism without intricate interconnections. By the asymptotic analysis on energy efficiency, we demonstrated that our design can achieve a scalable energy efficiency while satisfying the high throughput requirement simultaneously. The experiments show that, by reducing the interconnection power, our proposed architecture sustain the energy efficiency for various throughput requirements. The proposed PMA scheme can be employed to reduce the memory power effectively. In the future, we plan to work on an accurate performance model for energy-efficiency estimation, which can be used for design space exploration to obtain a power optimized high throughput FFT design.

References

[1] L. R. Rabiner, B. Gold, and C. K. Yuen, “Theory and application of digital signal processing,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 2, pp. 146–146, 1978.

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

[3] 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.

[4] Y.-W. Lin, H.-Y. Liu, and C.-Y. Lee, “A 1-GS/s FFT/IFFT processor for UWB applications,” *IEEE Journal of Solid-State Circuits*, vol. 40, no. 8, pp. 1726–1735, 2005.

[5] J. W. Cooley and J. W. Tukey, “An algorithm for the machine calculation of complex Fourier series,” *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[6] B. Baas, “A low-power, high-performance, 1024-point FFT processor,” *IEEE Journal of Solid-State Circuits*, vol. 34, no. 3, pp. 380–387, 1999.

[7] C.-W. J. Wen-Chang Yeh, “High-speed and low-power split-radix FFT,” *IEEE Transactions on Signal Processing*, vol. 51, no. 3, pp. 864–874, 2003.

[8] E. H. Wold and A. M. Despain, “Pipeline and parallel-pipeline FFT processors for VLSI implementations,” *IEEE Transactions on Computers*, vol. 100, no. 5, pp. 414–426, 1984.

[9] S. Hong, S. Kim, M. Papaefthymiou, and W. Stark, “Power-complexity analysis of pipelined VLSI FFT architectures for low energy wireless communication applications,” in *Midwest Symposium on Circuits and Systems*, 1999, pp. 313–316.

[10] S. He and M. Torkelson, “A new approach to pipeline FFT processor,” in *Proceedings of IPPS’96*, pp. 766–770.

[11] S. Choi, R. Scrofano, V. K. Prasanna, and J.-W. Jang, “Energy-efficient signal processing using FPGAs,” in *Proceedings of FPGA ’03*, 2003, pp. 225–234.

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

[13] K. K. Poon, S. J. Wilton, and A. Yan, “A detailed power model for field-programmable gate arrays,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 10, no. 2, pp. 279–302, 2005.

[14] B. Chazelle and L. Monier, “A model of computation for VLSI with related complexity results,” in *Proceedings of ACM symposium on Theory of computing*, 1981, pp. 318–325.

[15] M. Püschel, P. A. Milder, and J. C. Hoe, “Permuting streaming data using rams,” *Journal of the ACM*, vol. 56, no. 2, pp. 10:1–10:34, 2009.