High throughput energy efficient multi-FFT architecture on FPGAs (Draft)

*Ren Chen, †Neungsoo Park, *Viktor K. Prasanna
*Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, USA 90089
Email: renchen, prasanna@usc.edu
†Department of Computer Science and Engineering
Konkuk University, Seoul, Korea
Email: neungsoo@konkuk.ac.kr

***Abstract*—To process high-rate streaming data, throughput is one of the key performance metrics for FFT design. However, high throughput FFT architectures consume large amount of power due to complex routing or excessive memory access. In this paper, we propose a Cooley-Turkey algorithm based, high throughput energy-efficient multi-FFT architecture. In the proposed architecture, we use multiple time-multiplexed pipeline FFT processors to achieve high throughput. Time-multiplexers instead of routing network are used for shuffling the intermediate data, thus reducing the burden of interconnection power for large FFT problem size. To reduce memory power, a method of dynamic memory activation is developed. For N-point FFT ($64 \leq N \leq 4096$), our designs improve the energy efficiency (defined as GOPS/Joule) by 17% to 26%, compared with a state-of-the-art design. The experimental results show that, for various throughput requirements, the proposed design can achieve $50 \sim 63$ GOPS/Joule, i.e. up to 78% of the estimated peak energy efficiency of FFT designs on FPGAs.**

***Index Terms* — Fast Fourier Transform, FPGA computing, High throughput FFT, Energy efficient design**

I. Introduction

As FPGAs are programmed specifically for the problem to be solved, they can achieve higher performance with lower power consumption than general purpose processors. Therefore, 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 multi-carrier modulation and demodulation in spread spectrum receiver [4]. In the design of high throughput FFT architectures, energy-efficient design techniques can be used to maximize performance under power dissipation constraints. The power consumption of an FFT implementation is highly dependent on the number of memory modules, processing units and the interconnection topology.

Fully spatial parallel FFT architecture, based on the complete Cooley-Turkey algorithm layout, is one of the potential high throughput designs [5]. It achieves high performance by using spatial parallelism, while requiring more routing resources. However, as the problem size grows, unfolding the architecture spatially is not feasible due to serious power and area issue brought by complex interconnections.

The pipeline FFT processors are a class of compact FFT hardware architectures which can compute the FFT sequentially based on the time-multiplexed technique [6], [7], [8]. This technique folds the FFT algorithm layout vertically, hence reducing area and resources. The computation units are reused by several different streams of data sequentially. Data inputs are continually fed through the processor and processed in a streaming manner. High computational performance per area can be achieved by these designs, while the throughput is reduced due to limited parallelism.

In this paper, we propose our architecture based on hybrid designs using time-multiplexing and spatial parallelization. Multiple time-multiplexed pipeline FFT processors are used to increase the task parallelism to meet a throughput requirement. For each processor, partially spatial parallelization is employed to increase the throughput. Compared with the fully spatial parallel architecture, our design effectively reduces the interconnection power. To reduce the memory power, dynamic memory activation is developed. We also make a high-level power analysis to estimate the energy efficiency for demonstrating the improved performance in our design. The experiment results show that the proposed design can achieve high energy efficiency for a given throughput target. Our contributions in this paper are summarized as follows:

1) A high throughput energy efficient multi-FFT architecture using time-multiplexing technique and spatial parallelization (Section III-A).
2) Improved performance by increasing both task and spatial parallelism without requiring complex routing network (Section III-B).
3) A dynamic memory activation method that significantly reduces the power consumption of the memory (Section III-C).
4) Demonstrated the improved energy-efficiency by high-level power analysis, compared with the spatial parallel architecture (Section IV).
5) Significant energy efficiency improvement compared with the state of the art FFT design (Section V-E).

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

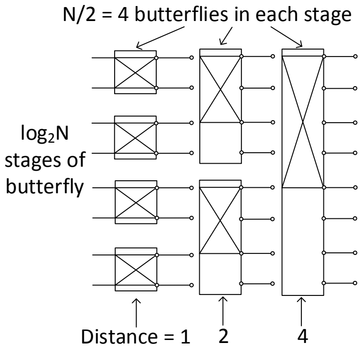


Fig. 1: The layout of 8-point fully spatial parallel FFT

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 high-level energy efficiency analysis. 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:

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

Radix-x Cooley-Tukey FFT is a well-known decomposition based algorithm for N-point DFT [3]. The Fig. 1 shows the complete layout of the 8-point fully spatial parallel FFT, which is based on radix-2 decimation in time algorithm. There are $\log_2 N$ stages of butterflies for N-point FFT. Complex routing network is required for data shuffling between the stages. As the wire length in the routing network increases with the problem size, the interconnection power of the fully spatial parallel FFT becomes a serious design issue, as the problem size grows [9].

A time-multiplexed FFT processor can vertically fold the fully spatial parallel FFT layout, thus saving area and power by eliminating the complex interconnections at the expense of performance. The processor performs N-point FFT in N/x (based on radix-x FFT) cycles using $\log_x N$ stages of butterfly. Fig. 2 shows the architecture overview of the time-multiplexed processor for 8-point radix-2 FFT. As shown in this figure, streaming data is shuffled by time-multiplexer unit in each stage so that data is fed through butterflies in a correct order. The time-multiplexer unit used between stages, is composed of data buffers and multiplexers. It enables to reuse the butterflies in each stage by time-multiplexing. For N-point FFT, each butterfly is reused for N/x times sequentially.

The time-multiplexed FFT processor is compact and power efficient, however the throughput is limited. In this paper, by appropriately expanding this design, multiple time-multiplexed processors are used to perform high throughput FFT computation without complex routing network.

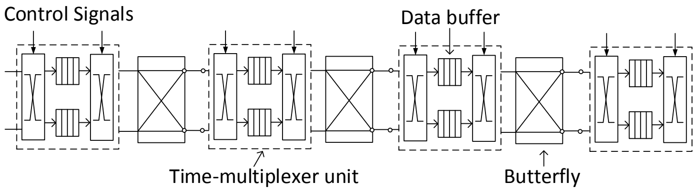


Fig. 2: The time-multiplexed processor for 8-point FFT

B. Related work

In [6], the authors designed an energy-efficient 1024-point FFT processor. Cache-based FFT algorithm was proposed to achieve low power and high performance. Energy-time performance metric was evaluated with different processor parameters. In [7], a high-speed and low-power FFT architecture was presented. They proposed a delay balanced pipeline architecture based on split-radix algorithm. Algorithms for reducing computation complexity were explored. These works mainly focused on increasing the peak frequency rather than parallelism to achieve higher 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 can achieve high performance per unit area with single-path or multi-path pipelines, while the throughput is limited due to small-scale parallelism.

In [2], a parameterized soft core generator for high throughput DFT was developed. This generator can automatically produce an optimized design with user inputs on performance and resource constraints. However, the energy efficiency is not presented in this work. In [11], the author presented a parameterized energy efficient FFT architecture. Their design is optimized to achieve high energy efficiency by varying the chosen architecture parameters. Throughput is not discussed in their paper.

In this work, we proposed high throughput multi-FFT architecture using energy-efficient design techniques. The proposed architecture will be presented in Section III.

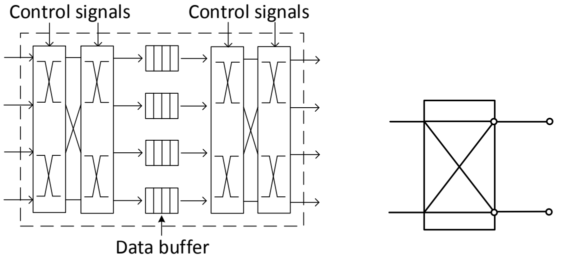
III. Architecture and Optimizations

A. Architecture building blocks

Our proposed architecture is based on the radix-x Cooley-Tukey FFT algorithm. The basic architecture is composed of multiple time-multiplexed pipeline FFT processors. Each processor consists of two types of building units (see Fig. 3): Butterfly unit and time-multiplexer unit. A complete design for a given throughput requirement can be obtained from combinations of the basic blocks.

- *Time-multiplexer unit*

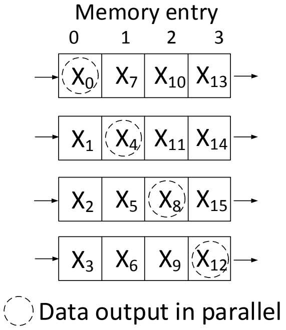
The time-multiplexer unit is composed of multiplexers and data buffers. The number of data buffers required for each time-multiplexer unit equals to the value of radix-x. The data inputs of this unit are produced by the former butterfly unit sequentially. After being time multiplexing, these data are



(a) Time-multiplexer unit

(b) Butterfly unit

Fig. 3: Architecture building blocks



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 16-point radix-4 FFT

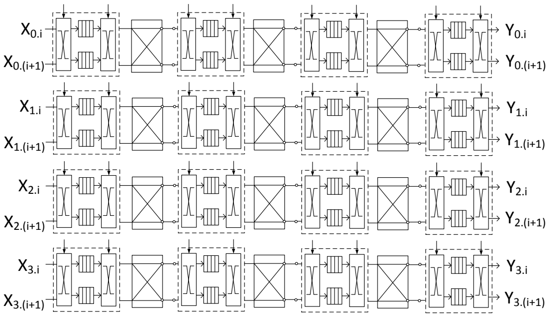


Fig. 5: High throughput 8-point 4-FFT architecture

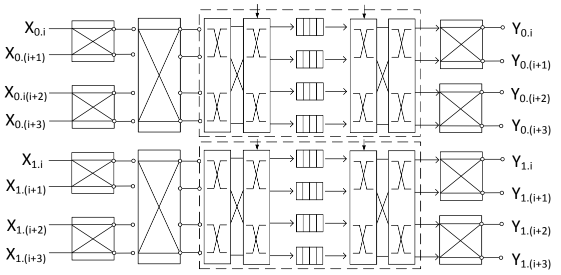


Fig. 6: High throughput 8-point 2-FFT architecture

then consumed by the subsequent butterfly in a different time sequence.

The time-multiplexer unit used for radix-4 FFT is shown in Fig. 3(a). In this figure, four data inputs are first permutated by the front multiplexers, and then temporarily stored in the four data buffers. After there have been N data entering into the data buffers, the data are then permutated by the back multiplexers and read out four by four in N/4 cycles.

Fig. 4 shows the data layout in the data buffers for 16-point radix-4 FFT. In the first cycle, four data inputs (X_0, X_1, X_2, X_3) are fed into the first entry of each data buffer without permutation. In the second cycle, another four data inputs (X_4, X_5, X_6, X_7) are written into the second entry of each data buffer with one location permutated. After the fourth cycle, 16 permutated data inputs are fed into the four data buffers. And the data inputs at the strides of four $(X_i, X_{i+4}, X_{i+8}, X_{(i+12)mod16}, i = 0, 1, 2, 3)$ are stored in different data buffers. After that, in each cycle, with alternating entries, the data at the strides of four are output four by four sequentially. As shown in Fig. 4, X_0, X_4, X_8, and X_{12} are firstly read out in parallel. For different FFT problem size N, the read and write addresses are generated with different strides, and the size of the data buffers is N/x.

- *Butterfly unit*

As shown in Fig. 3(b), the circles represent complex number operations. Butterfly computations and twiddle factor multiplications are completed in this unit. The twiddle factor coefficients are stored in the lookup tables.

As we use radix-x FFT, the value of x determines the numbers of adder/subtractors or multipliers needed. The butterfly unit in Fig. 3(b) is based on Radix-2 FFT, hence six real adder/subtractors and four real multipliers are needed to compute butterfly operations, and it takes two inputs and generates two outputs in parallel. Each input or output data contains real and imaginary components.

B. High throughput multi-FFT architectures

The proposed FFT architecture is composed of several independent parallel time-multiplexed processors. Each processor is pipelined and responsible for one N-point FFT task.

Decomposition based radix-x Cooley-Turkey FFT offers much flexibility to build a time-multiplexed processor. Three mapping parameters that characterize the decomposition-based N-point FFT algorithm are considered to generate our design:

- *Vertical parallelism (V_p)*: determines the number of parallel inputs of one time-multiplexed pipeline FFT processor. It also represents the spatial parallelism of one processor.
- *Number of FFT processors (N_p)*: determines the number of time-multiplexed pipeline FFT processors used in the architecture. As each processor is responsible for one FFT calculation task, N_p also represents the task parallelism.
- *Throughput requirement (Th)*: determines the number of sample points being computed in parallel in each clock cycle. $Th = V_p \times N_p$.

The three parameters above are chosen to create a specified design. Two different architectures for 8-point FFT are presented in the figures above. In Fig. 5, $V_p = 2$, $N_p = 4$, $Th = 8$ points per clock cycle, four time-multiplexed pipeline FFT processors are used. Each processor is responsible for one

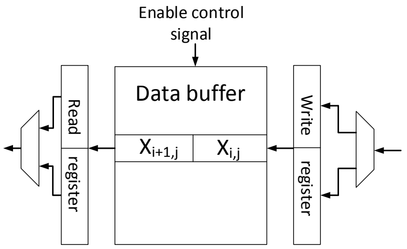


Fig. 7: Block diagram of data buffer using dynamic memory activation

FFT task. It takes four cycles to complete one task for each processor. This architecture achieves high throughput in the task level without complex routing network.

In Fig. 6, $V_p = 4$, $N_p = 2$, $Th = 8$ points per clock cycle, two time-multiplexed FFT processors are employed, and each processor is responsible for one task independently. This architecture achieves high throughput with higher spatial parallelism and requires less number of processors.

C. dynamic memory activation

Data buffer is one of the main components consuming large amount of power in our design. As data buffers are implemented in BRAMs on FPGAs, reducing power consumption of the BRAM makes a great impact on the power of the entire design. As described in Section III-A, the access pattern to the data buffer is regular. Data are read out or written into the data buffer with certain address strides one by one. The BRAM power can be reduced if we can turn off the BRAM periodically.

We propose a dynamic memory activation method to reduce power consumption of BRAM. Assuming the data width is one byte, and totally n bytes need to be read out from one data buffer sequentially in n clock cycles. The width and the height of the data buffer are one byte and n respectively. Then we reorganize BRAM as a memory of size $(n/m) \times m$-byte, and n is divisible by m. In this case, the width and the height of the new memory are exactly m-byte and n/m respectively. Fig. 7 shows the block diagram of the data buffer when $m = 2$. Two extra registers are required, one for the read operation and one for the write operation. Each register has the data width of two-byte. The read register is configured as parallel-in-serial-out shift register. In a read operation, this register reads in data of two-bytes from the BRAM, then output one-byte data per clock cycle. During the shift time, the BRAM is turned off. On the other hand, the write register is configured as serial-in-parallel-out shift register. For every clock cycle, the write register shifts the coming word into its parallel register. In every two cycles, the two-byte output is written into the BRAM at the second cycle. Note that the BRAM is turned off at the first cycle.

IV. High-level energy efficiency analysis

To demonstrate the energy efficiency improvement in our design, we perform the high-level power analysis based on basic assumptions and estimate the potential energy efficiency for performance comparison.

A. Notations and assumptions

Three major types of power are considered: *Computation power*, *Interconnection power*, and *Memory power*. Note that the I/O power is not considered as we assume only on-chip I/O ports are used and reducing I/O power is not our focus in this paper.

Assuming $F(N)$ as power consumption of a N-point FFT design, then $F(N)$ can be denoted as

$$F(N) = \Sigma_i f_i(N)(i = 1, 2, 3) \tag{2}$$

Here *computation power*, *interconnection power*, and *memory power* are respectively denoted as $f_1(N), f_2(N)$ and $f_3(N)$. Based on related theoretical foundations, we propose basic assumptions below for power analysis:

1) *Assumption 1:* Assuming the same operating frequency f for different designs.
2) *Assumption 2:* $f_1(N) \propto N_c$, N_c represents the average number of activated arithmetic units and DSP slices per clock cycle [12].
3) *Assumption 3:* $f_2(N) \propto l$, l means wire length [9].
4) *Assumption 4:* $f_3(N) \propto N_m$, N_m is the average number of activated memory slices per clock cycle.

Assumption 1 is true when the same target platforms and hardware implementation techniques are used. *Assumption 2* and *Assumption 3* are based on the papers referred. *Assumption 4* is true when using well organized memory block, such as BRAM on FPGAs. The power consumption of BRAM blocks will be analyzed in Section IV-B.

Assuming $E(N)$ as energy efficiency (defined as number of operations per Joul) of a design, $R(N)$ as number of operations(real number) per FFT calculation, and T as time required by $R(N)$ operations. Then $E(N)$ can be denoted as:

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

For N-point FFT, $R(N)$ is $O(N \log N)$.

B. Power analysis

In this part, based on the assumptions above, we theoretically calculate $F(N)$ of two target architectures, i.e. the fully spatial parallel architecture and our proposed architecture.

- *Computation power*

As there are $N \log N$ complex number operations in N-point FFT, N_c need to be $O(N \log N)$ for the two architectures. Based on the *Assumption 2*, the *computation power* $f_1(N)$ should be $O(N \log N)$ as the power of arithmetic unit or DPS slice does not increase with N.

- *Interconnection power*

As shown in Section II-A, for the fully spatial parallel N-point FFT architecture, there are $\log_2 N$ stages of interconnections, and the max wire length is $O(N)$ based on VLSI computational model. The total wire length l of the FFT network can be computed as $\alpha N[2 + 4 + ... + N/2] = O(N^2)$ [13]. In our proposed architecture, each of the $O(N)$ processors has $\log_2 N$ stages. In each stage, the wire length is constant, thus l is $O(N \log N)$. Based on the *Assumption 3*, the *interconnection*

TABLE I: Energy efficiency analysis

	Computation power ($f_1(N)$)	Interconnection power ($f_2(N)$	Memory power ($f_3(N)$	Energy efficiency $E(N)$
Fully spatial parallelized arch.	$O(N\log N)$	$O(N^2)$	$O(N\log N)$	$\frac{O(N\log N)}{O(N^2)}$
Our proposed arch.	$O(N\log N)$	$O(N\log N)$	$O(N\log N)$	$\frac{O(N\log N)}{O(N\log N)}$

power $f_2(N)$ can be computed as $O(N^2)$ and $O(N\log N)$ respectively for the fully spatial parallel architecture and our design.

- *Memory power*

Two types of memories, including Distributed RAM and Block RAM (BRAM), exist on FPGA. Both types of memories maybe used in our applications. As BRAM is highly optimized in area, it consumes less power per bit than Distributed RAM. Hence we assume only BRAM is used. A BRAM slice (on Xilinx devices) is organized with a 36 Kb block containing two independent 18 Kb blocks. Despite how small the amount of memory required, a BRAM block has to be assigned to serve the purpose. Therefore, BRAM power is determined by the number of blocks used rather than the total size of memory [14], which matches with the *Assumption 4*. In our proposed architecture, each of the $O(N)$ processors has $\log N$ stages. In each stage, there are constant number of data buffers used, thus the total number of data buffers implemented in BRAM (N_m) is $O(N\log N)$. Note that stage pipeline registers instead of data buffers are needed for fully spatial parallel architecture, hence its N_m is also $O(N\log N)$. The memory power can be computed as

$$f_3(N) = N_m \times P_{access} = O(N\log N) \times P_{access} \tag{4}$$

P_{access} represents the average power per access of a BRAM slice, which is almost not affected by N. P_{access} for BRAM is determined by the factors such as operating frequency, duty cycle, access rate, and bit width of read out data. This indicates that the average BRAM power per access can be reduced by decreasing the access rate, thus reducing BRAM power. The dynamic memory activation presented in Section III-C is proposed based on this observation.

C. Energy efficiency analysis

Using the energy efficiency definition in Section IV-A, we performed the energy efficiency analysis of the two target architectures in this part. The energy efficiency is calculated based on Equation (3). $R(N)$ has been computed as $O(N\log N)$. T is a constant for both of the two architectures. Table I shows the energy efficiency comparison between the two target architectures. It shows that the energy efficiency of the fully spatial parallel architecture is $O(N\log N)/O(N^2)$, which dramatically decreases as problem size grows. For our proposed architecture, the energy efficiency is $O(N\log N)/O(N\log N)$, which is almost not affected by the increasing problem size. This indicates that when problem size N and throughput Th increases, our design can be optimized to sustain a high energy efficiency.

V. Experimental results and analysis

In this section, we compare the performance result of our proposed architecture with both the fully spatial parallel architecture and a state-of-the-art design.

A. Experimental Setup

All the designs are implemented in Verilog on Virtex-7 FPGA (XC7VX980T, speed grade -2L) using Xilinx ISE 14.4. Inputs are 16-bit fixed point numbers. The designs were verified by post place-and-route simulation. The input test vectors for the simulation were randomly generated and had an average switching activity of 50%. All the reported results are post place-and-route results. We used the VCD file (value change dump file) as input to Xilinx XPower Analyzer to produce accurate power dissipation estimation [14]. The operating frequency of all the evaluated designs are set to be 333 MHz for power evaluation. *Energy efficiency* (defined in Section IV-A) and *Throughput* (defined in Section III-B) are considered as two performance metrics in this paper.

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 for the processing element and ideal conditions. Thus, we ignore all overheads such as memory energy dissipation, I/O, access to memory, cache and other 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. The other components such as routing, memory are not considered. We still keep the operating frequency as 333 MHz and using radix-4 FFT for consistency. For radix-4 FFT, the number of adders and multipliers required are $\frac{11}{4}N\log_2 N$ and $\frac{3}{2}N\log_2 N$ respectively. We performed the power estimation of the "minimal" architecture by post place-and-route simulation. Based on Equation (3), the peak energy efficiency is 81 GOPS/Joule for radix-4 FFT.

We use the peak energy efficiency as an upper bound on the performance of the chosen algorithm and architecture for FFT and compare the sustained performance of an implementation against this bound. Note that as the multiplier and adder cores improve, we can expect a corresponding increase in the sustained performance of our algorithm and the architecture for FFT.

C. Energy efficiency and power evaluation

In this section, we present the energy efficiency and power comparisons of the fully spatial parallel architecture and our proposed architecture by varying FFT problem size N. In this experiment, N varies from 64 to 4096, and V_p is chosen to be 4 in our design. We keep the same throughput, which increases with N, for the two architectures.

The energy efficiency results of the two architectures are presented in Fig. 9. As shown in this figure, when $N = 64$, the fully spatial parallel architecture achieves higher energy

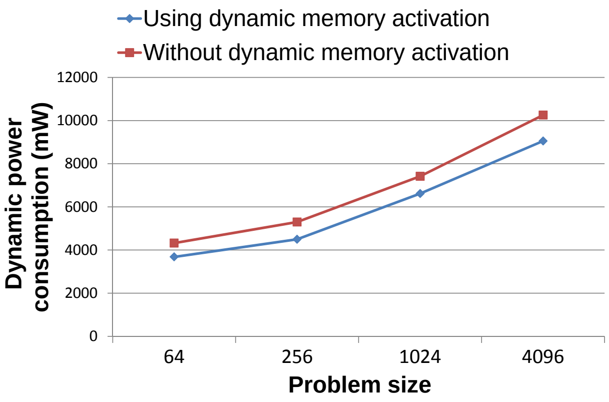


Fig. 8: Power evaluation with varying problem size for dynamic memory activation

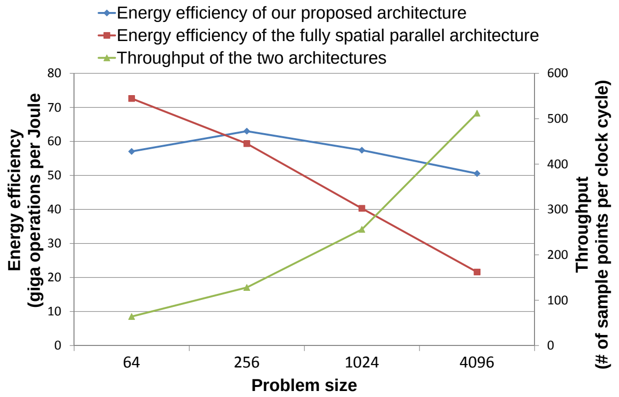


Fig. 9: Energy efficiency evaluation with varying problem size and throughput

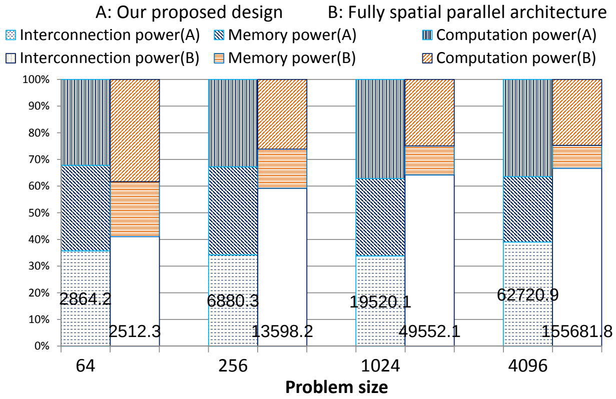


Fig. 10: Power evaluation with varying problem size for our proposed design and the fully spatial parallel architecture

efficiency than our design. The reason for that is the interconnection power is not dominant in the entire power consumption when N is small. However, when N continually grows, the energy efficiency of the fully spatial parallel architecture significantly decreases, which matches with our high-level analysis in Section IV-C. The experimental results also show that, when N increases, with only a slight performance decline, our proposed design can achieve $50 \sim 63$ GOPS/Joule. This result also conforms to the previous conclusion by high-level power analysis. Compared to the upper bound on energy efficiency of FFT (discussed in V-B), our designs achieve up to 78% of the estimated peak energy efficiency $(81 GOPS/Joule)$. This result indicates that, with reducing the complex routing, our design sustain high percentage of the upper bound on energy efficiency of FFT.

Fig. 10 shows the percentage of the three major power consumption of the two architectures with varying N. From the experimental results, the *interconnection power* increases significantly from 41% to 66% when N increases. This result shows a similar power increasing tendency as we analyzed in Section IV-B. However, when N is very small, such as $N \leq 64$, the two architectures consume similar amount of interconnection power. It indicates that appropriately increasing spatial parallelism can be feasible to achieve higher energy efficiency for small values of N. For our design, the interconnection power accounts for $33\% \sim 38\%$ of the entire power for various N, i.e. only increases within a narrow range in terms of percentage.

D. Evaluation of dynamic memory activation

In this section, we evaluate the power optimization results of our design using the proposed dynamic memory activation. We observe that power consumption can be effectively reduced by using this technique when the read/write buffer size is chosen to be 2. Fig. 8 shows the power reduction results after using dynamic memory activation for our proposed design with varying problem size N. N varies from 64 to 4096. V_p and N_p are chosen to be 4 and 8 respectively. The results show that the entire dynamic power consumption can be reduced by 10% to 15% for various problem size N. It indicates that this method can be used to reduce the entire power effectively for different throughput requirements. The reason for that is, each data buffer in our design can be applied with this method, thus the power reduced by dynamic memory activation also increases with problem size N.

E. Performance comparison

We finally use SPIRAL FFT IP core to compare with our proposed architecture. The SPIRAL DFT/FFT IP Generator can automatically generate customized DFT soft IP cores in synthesizable RTL Verilog with user inputs [2]. The available parameters of the DFT core generator include transform size, data precision, etc.

In this comparison, we use $V_p = 4$, $N_p = 8$ in our design. And we keep the same throughput $(Th = 32)$ for the two architectures for various N, as the SPIRAL FFT IP cores support Th to be set as up to 64. For the design from SPIRAL, the codes of N-point (16-bit fixed point) FFT are automatically generated by the SPIRAL Core generator. The architecture

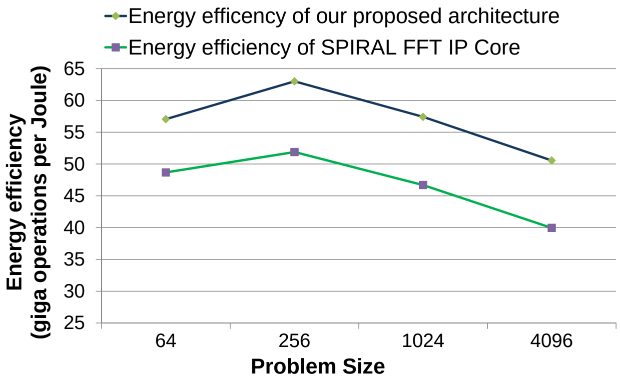


Fig. 11: Comparison between the proposed architecture and the SPIRAL FFT IP Cores for energy efficiency ($V_p = 4$, $N_p = 8$, $Th = 32$)

is fully streaming and the data are presented in their natural ordering. As shown in Fig. 11, our proposed design improves energy-efficiency by 17% to 26%, compared with the SPIRAL FFT IP Cores. Note that in this experiment, we still consider using on-chip I/O ports for both designs.

VI. Conclusion

In this work, we presented a high-throughput energy-efficient architecture using Radix-x Cooley-Tukey FFT algorithm. The proposed designs achieve high throughput by increasing both task and spatial parallelism without complex interconnections. We demonstrated the improved energy efficiency by high-level power analysis for our design. The experiments show that, by reducing the interconnection power, our proposed architecture could sustain the energy efficiency for various throughput requirements. The proposed dynamic memory activation method can be employed to reduce memory power effectively for various problem size, hence improving the energy efficiency. In the future, we plan to work on an accurate high-level performance model for energy-efficiency estimation, which can be used for design space exploration to obtain an energy efficiency optimized high throughput FFT design.

References

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

[2] G. Nordin, P. Milder, J. Hoe, and M. Puschel, “Automatic generation of customized discrete Fourier transform IPs,” in *Design Automation Conference, 2005. Proceedings. 42nd*, 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 *Circuits and Systems, 1999. 42nd Midwest Symposium on*, vol. 1, 1999, pp. 313–316 vol. 1.

[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 the 2003 FPGA*, pp. 225–234.

[12] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, “Power modeling and characteristics of field programmable gate arrays,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 11, pp. 1712–1724, 2005.

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

[14] “XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices,” http://www.xilinx.com/support/documentation.