FPGA ARCHITECTURE FOR 2D DISCRETE FOURIER TRANSFORM BASED ON 2D DECOMPOSITION FOR LARGE-SIZED DATA

Jung Sub Kim, Chi-Li Yu[†]*, Lanping Deng*[†]*, Srinidhi Kestur,*

Vijaykrishnan Narayanan and Chaitali Chakrabarti[†]

Dept. of Computer Science and Engineering, Pennsylvania State University
email: {jskim, kesturvy, vijay}@cse.psu.edu
[†]: School of Electrical, Computer and Energy Engineering, Arizona State University
email: {chi-li.yu, ldeng2, chaitali}@asu.edu

ABSTRACT

Applications based on Discrete Fourier Transforms (DFT) are extensively used in various areas of signal and digital image processing. Of particular interest is the two-dimensional (2D) DFT which is more computation- and bandwidth-intensive than the one-dimensional (1D) DFT. Traditionally, a 2D DFT is computed using Row-Column (RC) decomposition, where 1D DFTs are computed along the rows followed by 1D DFTs along the columns. Both application specific and reconfigurable hardware have been used for high-performance implementations of 2D DFT. However, architectures based on RC decomposition are not efficient for large input size data due to memory bandwidth constraints. In this paper, we propose an efficient architecture to implement the 2D DFT for large-sized input data based on a novel 2D decomposition algorithm. This architecture achieves very high throughput by exploiting the inherent parallelism due to the algorithm decomposition and by utilizing the row-wise burst access pattern of the external memory. A high throughput memory interface has been designed to enable maximum utilization of the memory bandwidth. In addition, an automatic system generator is provided for mapping this architecture onto a reconfigurable platform of Xilinx Virtex5 devices. For a $2K \times 2K$ input size, the proposed architecture is 1.96x times faster than RC decomposition based implementation under the same memory constraints, and also outperforms other existing implementations.

1. INTRODUCTION

Discrete Fourier Transform (DFT) is widely used in digital signal processing (DSP) and scientific computing applications. In particular, the two-dimensional (2D) DFT is used in a wide variety of imaging applications which need spectral and frequency-domain analysis, such as watermarking, finger print recognition, oral surgery and radiology. The image sizes of many of the applications have increased over the years. In synthetic aperture radar (SAR) processing [1], digital holographic imaging [2] and medical imaging, for instance, the required data size could be as large as 2048×2048. Therefore, there is a need for new algorithms and architectures to support 2D DFT of large data sizes.

Most of the existing 2D DFT algorithms have been primarily based on Row-Column (RC) decomposition where the 2D DFT is computed by successively applying 1D DFT along rows and then along columns. Existing 2D DFT implementations include software solutions, such as FFTW [3] and Spiral [4], which can run on general purpose processors, or parallel/vector processors and supercomputers. Though these platforms can achieve high performance, they are expensive and not suitable for embedded applications.

There are also many hardware solutions. Several of them are based on the Fast Fourier Transform (FFT) [5]. These include the dedicated FFT processor chips [6] [7], DSP processor [8] [9], and field programmable gate array (FPGA) based implementations [2] [10]. We concentrate on FPGA-based architectures since they can be reconfigured according to user-specified design parameters and offer flexibility while maintaining high performance. Moreover, FPGAs are being widely used in various embedded signal and imaging processing systems such as smart cameras, radar image reconstruction in which FFT module is a key component. Consequently, the exploration of FFT modules in FPGAs is the focus of this work.

In this paper, we describe architectures for 2D DFT that are targeted for large data sizes. In other FPGA architectures, such as those in [10], the performance degrades significantly when data size increases and the data does not fit in the on-chip memory. The bottleneck is the data transfer between the off-chip and on-chip memories. This problem has been addressed in [2], but it requires an additional transpose operation. We avoid it by implementing a new two-dimensional (2D) decomposition algorithm, and designing a customized memory interface which maximizes the external memory bandwidth. The proposed algorithm partitions the original data into a mesh of sub-blocks, performs data exchange operation between sub-blocks and then computes 2D DFT on the sub-blocks. The size of the sub-blocks is a function of the FPGA resources. The experimental results demonstrate that our architecture based on 2D decomposition achieves better performance than optimized architectures based on RC decomposition.

The rest of the paper is organized as follows. Section 2 briefly introduces 1D and 2D DFT and derives the proposed 2D decomposition algorithm. Section 3 describes in detail our novel FPGA architecture for 2D DFT. Section 4 describes the automatic system generator. Various configurations of our architecture are evaluated in Section 5, and concluding remarks are given in Section 6.

2. 2-DIMENSIONAL DECOMPOSITION FOR 2D-DFT

In this section, the decomposition of 1D DFT is described in Section 2.1, followed by the 2D decomposition algorithm for 2D DFT in Section 2.2 and the functional components of the 2D DFT in Section 2.3.

2.1. Decomposition of 1D DFT

A 1D DFT of length N can be decomposed and computed by a series of smaller transforms and permutations. We first represent DFT in the matrix-vector multiplication form as

$$[X_0\, X_1 \ldots\, X_{N-1}]^T = F_N \cdot [\, x_0\, x_1 \ldots\, x_{N-1}]^T \tag{1}$$

where F_N is the twiddle factor matrix.

The decomposition of 1D DFT is essentially representation of F_N as a product of sparse matrices and is described as follows [11], [12], [13].

$$F_N = P_{N,p}(I_p \otimes F_m)\tilde{D}_N(F_p \otimes I_m) \tag{2}$$

where $N = p \cdot m$. p and m are both integers, and I_m is the $m \times m$ identity matrix. $\tilde{D}_N$ is a diagonal matrix of twiddle factors, and $\otimes$ is the Kronecker or tensor product and can be expressed as

$$\tilde{D}_N(j,j) = W_N^{(j \bmod m)\cdot\lfloor j/m \rfloor} \text{ for } j = 0, 1 \ldots N-1 \tag{3}$$

$$A_n \otimes B_m = [a_{k,l}B_m]_{0\le k,l<n} \text{ for } A = [a_{k,l}]_{0\le k,l<n} \tag{4}$$

Finally, $P_{N,p}$ denotes the permutation with stride p.

The traditional radix-2 FFT can be considered a specific case of recursive decomposition with factor $p = 2$.

2.2. 2D decomposition algorithm

The general form of 2D DFT is described in matrix form as follows:

$$Y = F_M \cdot X \cdot F_N \tag{5}$$

where input X and output Y are of size $M \times N$, and F_M and F_N are DFT matrices.

The expression $(F_M \cdot X)$ is traditionally calculated by applying an M-point DFT for each column of X. As described in Section 2.1, an M-point DFT can be replaced by the sparse matrix product form as depicted in Eq. (2). Hence, by partitioning a column of size M into p sub-blocks, the expression $(F_M \cdot X)$ can be written as follows:

$$F_M \cdot X = P_{M,p} \cdot (I_p \otimes F_{M/p})\tilde{D}_M(F_p \otimes I_{M/p}) \cdot X \tag{6}$$

where the permutation $P_{M,p}$ term in Eq. (2) is taken into account in the final stage.

Similarly, the expression $(X \cdot F_N)$ in Eq. (5) can be written as follows:

$$X \cdot F_N = X \cdot (F_q \otimes I_{N/q})\tilde{D}_N(I_q \otimes F_{N/q}) \cdot P_{N,q} \tag{7}$$

Extending such a partitioning to both row-wise and column-wise elements, the 2D decomposition of Eq. (5) on a $p \times q$ mesh is written as follows:

$$\tilde{Y} = (I_p \otimes F_{M/p})\tilde{D}_M(F_p \otimes I_{M/p}) \cdot X \cdot (F_q \otimes I_{N/q})\tilde{D}_N(I_q \otimes F_{N/q}) \tag{8}$$

Y is obtained by applying a bit-reverse permutation (P) on $\tilde{Y}$ of Eq. (8).

2.3. Functional components of the 2D decomposition algorithm

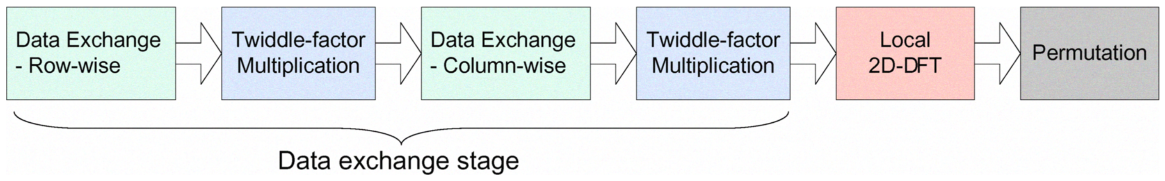
Eq. (9) pictorially demonstrates the sequence of operations involved in the computation of the decomposed 2D-DFT.

$$Y = P(\underbrace{(I_p \otimes F_{M/p}) \underbrace{\tilde{D}_M(F_p \otimes I_{M/p}) \cdot \underbrace{X \cdot (F_q \otimes I_{N/q})\tilde{D}_N}_{\text{Step 1}}}_{\text{Step 2}}(I_q \otimes F_{N/q})}_{\text{Step 3}}) \tag{9}$$

Input X of size $M \times N$, is partitioned into a $p \times q$ mesh where each sub-block in the mesh is of size $M/p \times N/q$. The four main steps are described below Figure 1 presents the functional flow of the proposed 2D decomposition algorithm.

Step 1. Row-wise data exchange with twiddle-factor multiplication:

There are q sub-blocks in each row, and each element inside one sub-block has to do data exchange with the corresponding elements in the other $(q-1)$ sub-blocks. This data exchange (DX) operation can be implemented as a q-point 1D FFT followed by a twiddle-factor multiplication (with $\tilde{D}_N$). Since there are $M/p \cdot N/q$ elements in each sub-block, there are $(MN/pq)(q \cdot \log_2 q)$ arithmetic operations with q-point 1D FFT for each row, and $(MN/pq)(q \cdot \log_2 q)p$ for all rows. Including MN operations for twiddle-factor multiplication $(\tilde{D}_N)$, it takes a total of $(MN/pq)(q \cdot \log_2 q)p + MN \approx O[MN(1 + \log_2 q)]$ arithmetic operations. Note that all q-point 1D FFTs can be computed in parallel.



Data Exchange - Row-wise
Twiddle-factor Multiplication
Data Exchange - Column-wise
Twiddle-factor Multiplication
Local 2D-DFT
Permutation
Data exchange stage

Fig. 1. The functional flow graph of 2D DFT

Step 2. Column-wise data exchange with twiddle-factor multiplication:
$Y_2 = \tilde{D}_M(F_p \otimes I_{M/p}) \cdot Y_1$

Similar to Step 1, the DX is repeated for the p sub-blocks in each column. This step has a complexity of $O[MN(1 + \log_2 p)]$ operations.

Step 3. Local 2D DFT on each sub-block:
$\tilde{Y} = (I_p \otimes F_{M/p})Y_2(I_q \otimes F_{N/q})$

After the row-wise and column-wise data exchange with twiddle-factor multiplications, 2D FFT computation is performed on each sub-block of size $M/p \cdot N/q$. This operation is fully parallel for all sub-blocks, and only limited by resource constraints in the underlying architecture. No data communication is required between any sub-blocks.

Step 4. Output permutation:
$Y = P(\tilde{Y})$

A bit-reverse permutation is required before generating the output. This is done by the host computer before display.

Note that there are other 2D decomposition methods, such as Vector Radix FFT [14], which recursively decomposes the 2D DFT into small-sized ones, like 2 × 2 or 4 × 4 2D DFT. While this method can effectively reduce the number of multiplications, the recursive decomposition makes hardware implementation and memory addressing much more complicated.

3. PROPOSED 2D-DFT ARCHITECTURE

3.1. Architecture for 2D Decomposition

The input data is stored in the external memory, and is logically partitioned into a mesh of sub-blocks, as shown in Figure 2. During each step mentioned in Section 2, portions of data are loaded into the FPGA local memory, processed, and then stored back to the external memory. Note that only row-wise accesses to the external memory are adopted in the proposed design. The architecture and its operations are described in details as follows.

The data exchange stage consists of the first four blocks in Figure 1. In this stage, equal number of samples from the same position in each sub-block are loaded into local memory. Note that the data in the external memory are accessed only along the row direction as depicted in Figure 2. This pattern is especially advantageous for accessing a dynamic memory, such as DDR2-400 SDRAM, which only favors row-wise burst access. Then, both the row-wise and column-wise data exchange are performed by the processing-element (PE) array. The operations are repeated until the entire data is traversed, as shown in Figure 2.

In the local 2D DFT stage (corresponding to the fifth block in Figure 1), fixed number of contiguous sub-blocks of the 2D data are loaded into FPGA local memory and the PE array computes 1D transforms along rows and then along columns. This operation is repeated for all the blocks.

The PE array consists of multiple PEs, where each PE is formed by a primitive 1D FFT IP core and a complex multiplier. The primitive FFT is used for both data exchange and local 2D DFT. The complex multiplier is used for twiddle-factor multiplication during data exchange. All PEs operate in parallel and access data from/to multi-banked local memory simultaneously without conflicts. The choice of number of PEs and size of the primitive FFTs are explained in Section 4. Note that for ease of implementation, an input size of $N \times N$ is always assumed, where N is a power of 2, otherwise zero-padding is applied.

While the PE array can finish computations very fast, the bottleneck is the interface to external memory. Xilinx's Multi-Port Memory Controller (MPMC), for instance, is a very versatile controller supporting SDRAM/DDR/DDR2 memory. However, its peak throughput is only 50% of a DDR2-SDRAM DIMM's peak transfer rate. To alleviate this bottleneck, we design a customized high throughput memory interface.

3.2. High Throughput Memory Interface

The customized memory interface, as shown in Figure 3, has has a 128-bit wide internal data-bus. Since the DDR2 SDRAM device has an operating frequency of 200MHz and an user application in FPGA runs at 100MHz, the memory interface operates at 200MHz and has a 256-bit wide data bus between the interface and the application. This enables data transfer rates of up to 256 bits at 100 MHz or 3200 MBytes per sec. Together with double buffering technique on local memory, the customized memory interface enables us to completely overlap communication with computation and avoid any loss of performance due to communication bottle-neck. This memory interface can be ported onto any FPGA board with SDRAM DIMMs.

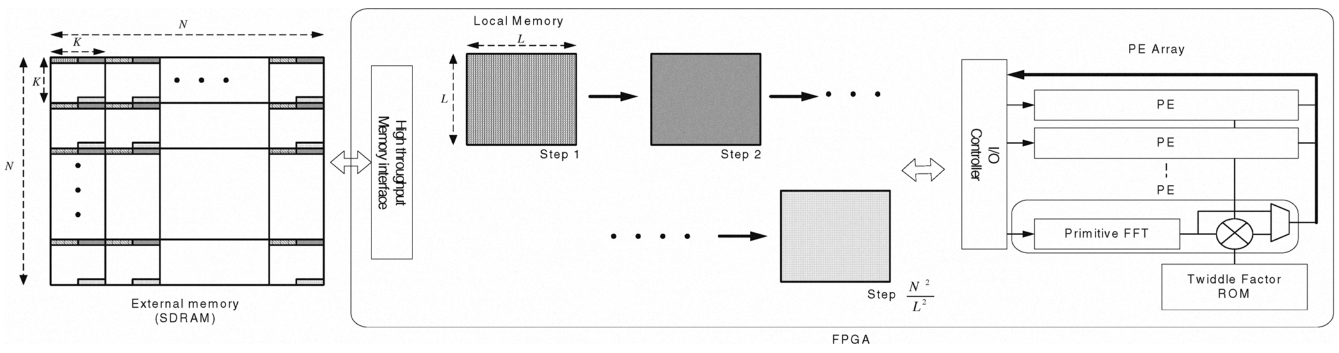


Fig. 2. Proposed 2D DFT architecture and Data Exchange access pattern from external memory

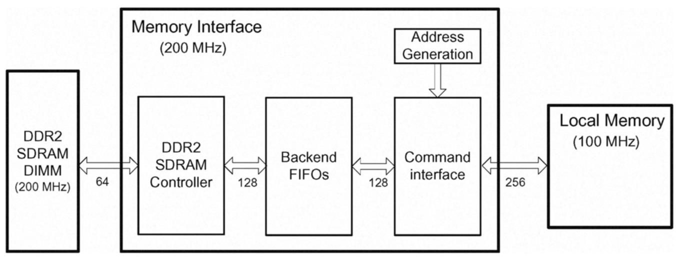


Fig. 3. High throughput memory interface

4. AUTOMATIC 2D DFT SYSTEM GENERATOR

Given the specification of input data, our system can automatically generate a 2D DFT implementation based on the proposed 2D decomposition algorithm. The automation flow is shown in Figure 5. A design optimizer determines the best decomposition based on input specifications, and passes this information to the system generator, which generates hardware module in Verilog or VHDL. Then, the HDL files are fed into the FPGA tool to produce final configuration bit-files. Note that no user intervention is required for the entire process.

The input specifications include data size, target device, memory type, and accuracy requirement. The data size is a power of two, typically from 512×512 to 4096×4096. The target FPGA platform is Virtex5 from Xilinx. For memory type, only DDR2 SDRAM is currently supported to provide large memory bandwidth. However, the memory interface is easy to extend to other memory types by replacing SDRAM controller block. The user can choose either 16-bit implementation for resource constrained designs, or 32-bit implementation for high accuracy designs.

The design optimizer is used to find the best partition for input data. While the sub-block size K can be any number such that N/K in an integer, the design optimizer sets K to be $\sqrt{N}$ or its closest power-of-two. By doing this, the maximum size of DFTs used in data exchange and local DFT can be minimized. Also, the same FFT IP core can be used for both data exchange and local DFTs and on-chip hardware resource

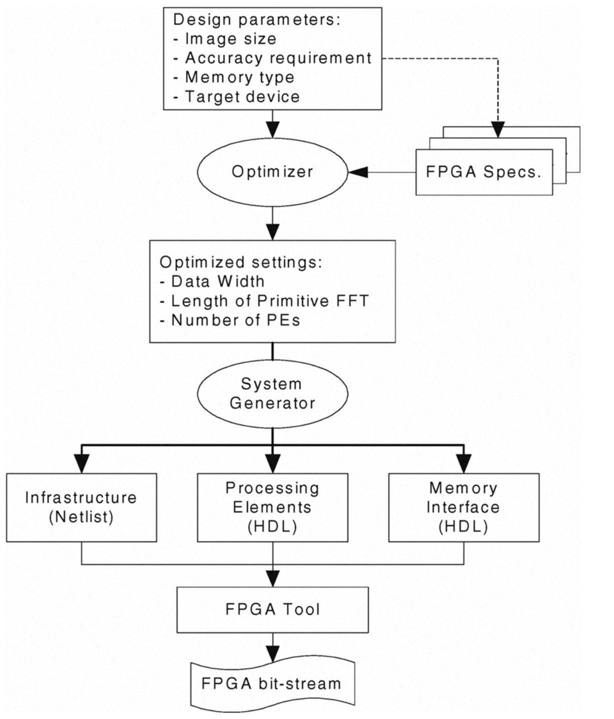


Fig. 4. Automation flow of generating architecture for 2D Decomposition based 2D DFT

can be saved.

The number of PEs is set to the number of image samples accessible from external memory per cycle. This is because Xilinx FFT IP can deal with 1 datum/cycle. So if the memory interface is 256 bits and data width of an image sample is 32 bits, we can read 8 samples from external memory to the FPGA and, therefore, the design optimizer will choose '8' as the number of PEs.

Once the design optimizer decides the primitive FFT and memory partitioning, the system generator generates all the required hardware modules. The modules are infrastructure block, FFT core block and memory interface block. The infrastructure blocks such as UART and host connection are fixed, whereas other blocks are user specified and provided in a template format, with several parameters that are set based on decisions of the design optimizer. After the generation of these modules, scripts for Xilinx flow are produced to run Xilinx tool automatically. Finally, the configuration bit file consisting of both hardware bit file and software binaries are generated from FPGA tool.

5. EVALUATION

In this section, the 2D DFT architecture generated using the DFT system generator is evaluated. First, the evaluation on high throughput memory interface is presented in Section 5.1. Then, the evaluation of performance for various input sizes are presented and compared with existing solutions in Section 5.2. For the evaluations, Xilinx 10.1 tool set and Modelsim 6.4 were used, and Virtex5FX device is considered as the candidate FPGA device.

5.1. Memory throughput for RC- and 2D decomposition-based architectures

The performance of the memory interface for read, write and read+write operations for various memory access patterns for an input size of 2048 × 2048 and local memory size of 128 × 128 is shown in Figure 5. The memory interface operates at 100MHz, and one DDR2-400 SDRAM DIMM is considered. The performance is measured in terms of the number of cycles taken to read/write data from/to DDR2 SDRAM to/from local memory. It can be observed that column access is much slower compared to other access patterns, and hence it would be the bottle-neck for direct RC method. On the contrary, data exchange (DX) and local DFT (LFFT) are both row-wise accesses. Therefore, the proposed 2D DFT avoids the performance penalty caused by column access and hence can achieve a higher performance.

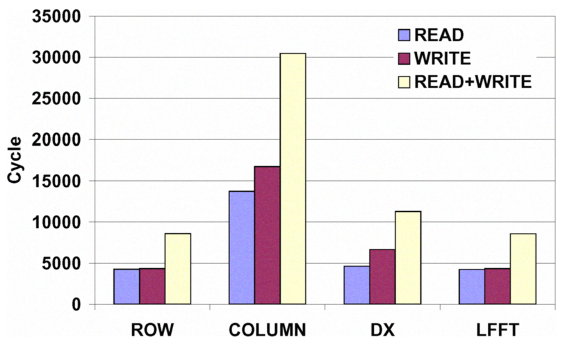


Fig. 5. Memory access time for different access patterns for on-chip memory size of 128 × 128

5.2. Comparison of performance

Table 1 shows the performance of the 2D DFT architecture for different input sizes and different architecture configurations. The performance is specified in terms of frame rate, which refers to the number of frames of input data that the 2D DFT architecture can process in a second. The size of the FFT primitive is chosen by the optimal partitioning strategy for the input data. It can be seen that the performance is inversely proportional to the input data size, but directly proportional to the number of PEs.

Table 1. Performance for different input sizes

Input size	Primitive	# of PE	Computation time(ms)	Frame rates
1024 × 1024	32PT FFT	8	5.4	182
		4	10.7	92
2048 × 2048	64PT FFT	8	22.4	44
		4	43.4	23
4096 × 4096	64PT FFT	8	90.4	11
		4	174.3	5

Table 2 compares the performance in terms of computation time and frame rate for FPGA implementations corresponding to (i) RC decomposition, (ii) Transpose (row-FFT - transpose - row-FFT) and (iii) the proposed 2D decomposition algorithm. This comparison is done for identical architecture constraints such one DDR2-400 SDRAM DIMM, 128 × 128 local memory size, 16-bit implementation and for an input data size of 2048 × 2048. The evaluation for the candidate architectures is performed using the proposed high throughput memory interface and under the assumption that the processing elements in FPGA support full performance that matches with the communication time, so that the performance is limited by the communication time. The table shows that under identical conditions, the 2D decomposition architecture provides a 96.7% improvement compared to the RC decomposition architecture and a 35.6% improvement compared to the transpose-based architecture.

Table 2. Performance comparison for different architectures for input size 2048 × 2048

	RC	Transpose	2D-DEC
Computation time (ms)	99.98	68.91	50.82
Frame rates	10	14	19

Further, Table 3 presents a comparison of other existing 2D FFT implementations with the proposed 2D decomposition based DFT(2D-DEC) architecture that utilizes maximum memory bandwidth on Virtex5 FPGA. Uzun et al. [15] have used SRAM as the external memory with VirtexE FPGA. Our 2D-DEC architecture provides significant performance improvement for the same input data size of 1024 × 1024. Dillon [16] has used multiple pipelined Virtex2 FPGAs for row-FFTs and column-FFTs and includes huge SRAM as intermediate memory for an input size of 2048 × 2048. In contrast, our 2D-DEC architecture uses a single FPGA with cheaper DDR2 SDRAM as external memory and so has lower performance. Lenart et al. [2] have implemented a transpose architecture with DDR SDRAM memory, but the performance is based on $0.13\mu m$ ASIC technology operating at 250 MHz. Our 2D-DEC architecture operates at 100 MHz on a Virtex5 FPGA and still provides performance improvement. Note that we would obtain similar improvement if implemented on a Virtex-II platform since the clock speed is also 100MHz.

Table 3. Performance comparison with existing works

Input size	Method	Frequency	Frame rate
1024 × 1024	Uzun [15]	27 MHz	13
	2D-DEC	100 MHz	182
2048 × 2048	Dillon [16]	125 MHz	120
	Lenart [2]	250 MHz	20
	2D-DEC	100 MHz	44

6. CONCLUSION

In this paper, we have proposed an efficient architecture to implement the 2D FFT for large input sizes based on a novel 2D decomposition algorithm. This architecture provides high performance by leveraging the inherent parallelism of the 2D decomposition and by scheduling data communication to overlap with computation. The memory bandwidth problem is alleviated by employing a custom-designed high throughput memory interface. In addition, a system generator is provided, which can automate the generation of an optimized version of the 2D FFT architecture for various input sizes. The evaluation of this architecture shows significant performance enhancements over existing 2D FFT implementations.

ACKNOWLEDGEMENTS

This work is supported in part by a grant from DARPA W911NF-05-1-0248. In addition, the authors gratefully acknowledge the help of Dr. Nikos Pitsianis and Dr. Xiaobai Sun of Duke University.

7. REFERENCES

[1] Y. K. Chan and S. Y. Lim, "Synthetic aperture radar (SAR) signal generation," *Progress In Electromagnetics Research B*, vol. 1, pp. 269–290, 2008.

[2] T. Lenart, M. Gustafsson, and V. Owall, "A hardware acceleration platform for digital holographic imaging," *Journal of Signal Processing System*, vol. 52, no. 3, pp. 297–311, September, 2008.

[3] M. Frigo and S. Johnson, "FFTW: An adaptive software of the FFT," *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1381–1384, 1998.

[4] M. Püschel and et al., "SPIRAL: Code Generation for DSP Transforms," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 232–275, Feb. 2005.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the machine computation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297–301, 1965.

[6] E. Bidet and et al., "A fast single-chip implementation of 8192 complex point FFT," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 300–305, Mar 1995.

[7] B. M. Baas, "A generalized Cached-FFT algorithm," *Proc. IEEE Int. conference on Acoustics, Speech, and Signal Processing*, vol. V, pp. 89–92, Mar. 2005.

[8] S. Berg and et al., "Critical review of programmable media processor architectures," *Proceedings of SPIE*, vol. 3655, pp. 147–156, 1998.

[9] C. Mermer, D. Kim, and Y. Kim, "Efficient 2D FFT implementation on mediaprocessors," *Parallel Computing*, vol. 29, no. 6, pp. 691–709, 2003.

[10] P. D'Alberto and et al., "Generating FPGA accelerated DFT libraries," *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 173–184, 2007.

[11] P. A. Milder and et al., "Discrete Fourier transform compiler: from mathematical representation to efficient hardware," Carnegie Mellon University, Tech. Rep. CSSI-07-01, 2007.

[12] C. Van Loan, *Computational framework of the fast Fourier transform*. SIAM, 1992.

[13] N. P. Pitsianis, "The Kronecker product in approximation and fast transform generation," Dissertation for the degree of Doctor of Philosophy, Cornell University, Jan. 1997.

[14] H. R. Wu and F. J. Paoloni, "The structure of vector radix Fast Fourier Transforms," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 9, September 1989.

[15] I. Uzun, A. Amira, and A. Bouridane, "FPGA implementations of fast Fourier transforms for real-time signal and image processing," *IEE Proceedings. Vision, Image, and Signal Processing*, vol. 152, no. 3, pp. 283–296, June 2005.

[16] T. Dillon, "Two Virtex-II FPGAs deliver fastest, cheapest, best high-performance image processing system," *Xilinx Xcell Journal*, vol. 41, pp. 70–73, 2001.