THOMSON REUTERS ENDNOTE

[Narayanam* *et al.*, 5(6): June, 2016] ISSN: 2277-9655
IC™ Value: 3.00 Impact Factor: 4.116

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

PAIRED FASTER FFT: GRIGORYAN FFT IMPLEMENTATION AND PERFORMANCE ON XILINX FPGAS AND TMS DSPS

Ranganadh Narayanam*, Artyom M. Grigoryan, Parimal A. Patel, Bindu Tushara D

DOI:

ABSTRACT

Discrete Fourier Transform is a principal mathematical method for the frequency analysis and has wide applications in Engineering and Sciences. Because the DFT is so ubiquitous, fast methods for computing DFT have been studied extensively, and continuous to be an active research. The way of splitting the DFT gives out various fast algorithms. In this paper, we present the implementation of two fast algorithms for the DFT for evaluating their performance. One of them is the popular radix-2 Cooley-Tukey fast Fourier transform algorithm (FFT) [1] and the other one is the Grigoryan FFT based on the splitting by the paired transform [2]. We evaluate the performance of these algorithms by implementing them on the Xilinx Virtex-II Pro [6], Virtex-4[9] and Virtex-5[7] FPGAs, by developing our own FFT processor architectures. We have evaluated the performances also by implementing on Texas Instruments fixed point DSP processors: TMS320C5416[17], TMS320C6748[17], TMS320C5515[17]. Finally we show that the Grigoryan FFT is working faster than the Cooley-Tukey FFT, consequently it is useful for higher sampling rates. Operating at higher sampling rates is a challenge in DSP applications. We proved that on Xilinx FPGAs and TMS DSPs, the Grigoryan FFT is performing at most 1.358 and 1.7 times faster than the Cooley-Tukey FFT respectively. We also confirm that for the same architectures Virtex-5 platform is better platform for implementing the Grigoryan FFT.

KEYWORDS: frequency analysis, fast algorithms, DFT, FFT, paired transforms, SIMULINK, CC Studio.

INTRODUCTION

In the recent decades DFT has been playing several important roles in advanced applications such as image compression and reconstruction in biomedical images, audiology research for analyzing biomedical brain-stem speech signals, sound filtering, data compression, partial differential equations, and multiplication of large integers. The fast algorithms for DFT always look for DFT process to be fast, accurate and simple. Fast is the most important [10]. FFT is universal in signal processing, but it can also be used to compress image and audio files, solve differential equations and price stock options, among other things.

Since the introduction of the fast Fourier transform (FFT), Fourier analysis has become one of the most frequently used tool in signal/image processing and communication systems; The main problem when calculating the transform relates to construction of the decomposition, namely, the transition to the short DFT's with minimal computational complexity. The computation of unitary transforms is complicated and time consuming process. Since the decomposition of the DFT is not unique, it is natural to ask how to manage splitting and how to obtain the fastest algorithm of the DFT. The difference between the lower bound of arithmetical operations and the complexity of fast transform algorithms shows that it is possible to obtain FFT algorithms of various speed [2]. One approach is to design efficient manageable split algorithms. Indeed, many algorithms make different assumptions about the transform length [2]. The signal/image processing related to engineering research becomes increasingly dependent on the development and implementation of the algorithms of orthogonal or non-orthogonal transforms and convolution operations in modern computer systems. The increasing importance of processing large vectors and parallel computing in many scientific and engineering applications require new ideas for designing super-efficient algorithms of the transforms and their implementations [2].

In the current age there are some algorithms and implementations that are coming for the Faster FFT, than Cooley-Tukey FFT for several different applications. One of them is "University of Michigan FFT algorithm": Anna Gilbert, Martin Strauss. The second one is "MIT FFT Algorithm": Dina Katabi, Piotr Indyk, Eric Price, Haitham Hassanieh. Ours is also one of them, it is the paired transform-based Grigoryan FFT which can be essentially applicable in military applications in Avionic and electronic warfare antennas, medical nano-robots and nano-enhanced reconnaissance and communication devices, where faster speed of FFT operation, requirement of some of the FFT coefficients earlier than waiting until the final stage of FFT; are highly useful. The Paired transform based Grigoryan FFT is a highly powerful tool.

In this paper we present the implementation techniques and their results for two different fast DFT algorithms. The difference between the algorithm development lies in the way the two algorithms use the splitting of the DFT. The two fast algorithms considered are radix-2 (Cooley-Tukey FFT) and paired transform [2] (Grigoryan FFT) algorithms. Implementation is done both on Xilinx FPGAs and Texas Instruments DSP processors. For FPGAs the modeling and simulations are done on SIMULINK of MATLAB with XSG: Xilinx System Generator (a high level visual tool for hardware generation). The implementation is done using the Xilinx project navigator backend software tools. We have developed specific C programs for TMS DSP processors using the Code Composer Studio Integrated Development Environment (CC Studio IDE), with implicit utilization of MAC engines.

Great speedups can be achieved for these algorithms by efficient implementation in dedicated hardware such as Application-Specific Integrated Circuits (ASICs). However, high "time-to market" has been a bottleneck for the ASICs. The evolution of Field Programmable Gate Arrays (FPGAs) along with high-level design tools such as from Altera, Xilinx System Generator have come as valuable and effective tool for high-level programmers to achieve better execution times in these reconfigurable hardware. The small time-to-market for FPGAs over VLSI models is the reason for popular choice of FPGAs in current market. FPGA expedite the time lag between hardware design and shipping time of the circuit from 2-3 years to a few weeks [16]. Advances in FPGA technology along with development of elaborate and efficient tools for modelling, simulation and synthesis have made FPGAs a highly useful platform. With a graphical environment based on SIMULINK and a pre- defined block set of Xilinx DSP cores, System Generator[14]-[15] meets the needs of both system architects who need to integrate the components of a complete design and hardware designers who need to optimize implementations. The salient features of FPGAs that make them superior in speed, over conventional general purpose hardware like Pentiums are their greater I/O bandwidth to local memory, pipelining, parallelism and availability of optimizing compiler [15]-[16]. Complex tasks, which involve, multiple image operators, run much faster on FPGAs than on Pentiums, in fact, some researches report an 800-time speed up by FPGA. There are several reasons for such large speed ups which FPGAs have over PCs. In comparison to an FPGA, hardware such as Pentium runs at memory speed, not at cache speed. So, even running at much higher clock frequency and having the facility of cache memory, it responds much slower than a comparable FPGA [16]. Frequency of operation in hardware such as Pentium can be increased up to a certain extent to increase the performance or the required data rate to process the image data, but increasing the frequency above certain limits causes system level and board level issues that become a bottleneck in the design. Considering all the advantages and to explore all these features we have chosen to implement on FPGAs. Implementing on DSP processors is a compulsory task for any DSP applications.

The implementation of the algorithms is done in Hardware point of view on the Xilinx Virtex-II Pro [6], Virext-4 [9], and Virtex-5 [7] FPGAs. Then we have implemented on Texas Instruments DSP Processors: TMS320C5416 [17], TMS320C6748 [17], TMS320C5515 [17]. The performance of the two algorithms is compared in terms of their sampling rates and also in terms of their hardware resource utilization.

The paper is organized in the following way. Section 2 presents the paired transform decomposition used in paired transform in the development of Grigoryan FFT. In Section 3 we present the implementation techniques for the radix-2 and paired transform algorithms on Xilinx FPGAs and TMS DSP processors. Section 4 presents the results. Finally with the Section 5 we conclude the work and further research.

DECOMPOSITION ALGORITHM OF THE FAST DFT USING PAIRED TRANSFORM

We consider the fast splitting of the discrete Fourier transform by the 1-D discrete paired transform (DPT) [1]-[4]. Let $\{x_n; n = 0(N-1)\}$ be the input signal of length $N > 1$. The N-point DFT of the signal x is defined as

$$X_p = (F_N{}^\circ x)_p = \sum_{n=0}^{N-1} x_n W^{np}, \quad p = 0:(N-1), \tag{1}$$

where the number $W = W_N = \exp(-2\pi i/N)$, i^2 = -1.This transform can be written in matrix form as $\mathbf{X} = [F_N]\mathbf{x}$, where **X** and **x** are column-vectors for X and x, and the matrix $[F_N] = \|W^{np}\|_{n,p=0:(N-1)}$. We consider the case of most interest, when the signal length is $N = 2^r, r > 1$. Unlike the Cooley-Tukey algorithm, on first stage of which F_N is calculated by two $F_{N/2}$, Grigoryan splits the transform F_N by $(r+1)$ short transforms as $\{F_{N/2}, F_{N/4}, F_{N/8}, \ldots, F_4, F_2, F_1, F_1\}$. Namely, the following matrix decomposition holds for the DFT:

$$[F_N] = \left[\left(\oplus_{k=0}^{r-1} [F_{N/2^{k+1}}]\right) \oplus 1\right] \cdot D_N \cdot [\chi'_N]$$

where $\oplus$ denotes the operation of the Kronecker sum of matrices and the diagonal matrix

$$D_N = \mathrm{diag}\{1, W, W^2, W^3, \ldots, W^{N/2-1}, W^2, W^4, \ldots, W^{N/2-2}, 1, W^4, W^8, \ldots, W^{N/2-4}, 1, \ldots, 1, 1\}.$$

The N-point unitary and binary discrete paired transform DPT $[\chi'_N]$ is described in the following way [2]. Given frequency-point $p \neq 0$ and time $t \in \{0, 1, \ldots, N-1\}$, let $\chi_{p,t}(n)$ be the function

$$\chi_{p,t}(n) = \begin{cases} 1, \text{if } np = t \bmod N, \\ 0, \text{ otherwise}, \end{cases} \qquad n = 0:(N-1) \tag{2}$$

The *2-paired*, or shortly the *paired* function is defined as

$$\chi'_{p,t}(n) = \chi_{p,t}(n) - \chi_{p,t+N/2}(n).$$

Here the integer $t \in \{0, 1, \ldots, N/2 - 1\}$ and $\chi'_{p,t+N/2}(n) = -\chi'_{p,t}(n)$.
The totality of the paired functions

$$\{\chi'_{2^n,2^n t}(n); n = 0:(r-1), t = 0:(N/2^{n+1} - 1), 1\}$$

is the complete and orthogonal set of functions of the paired transform χ'_N. Figure 1 shows two matrices of the 16- and 32-point pared transform χ'_N in parts a and b, respectively.

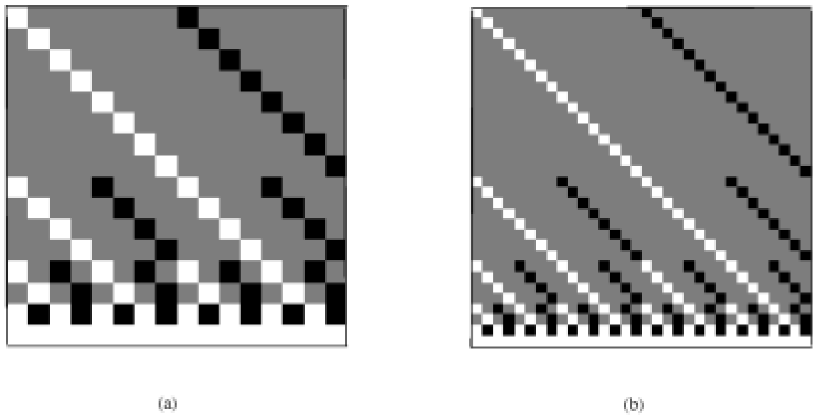


Figure 1. Three-level gray images of the paired matrices (a) 16×16 and (b) 32×32. (In these images, the color white is for coefficient 1, black is for −1, and gray is for 0.)

The number of operations of multiplication required to calculate the paired N-point FFT equals $M_N = N/2(\log_2 N - 3) + 2,\ r > 2$. The N-point discrete paired transform is fast and requires $(2N-2)$ operations of addition/subtraction. Figure 2 shows the signal-flow graph of the 8-point fast paired transform.

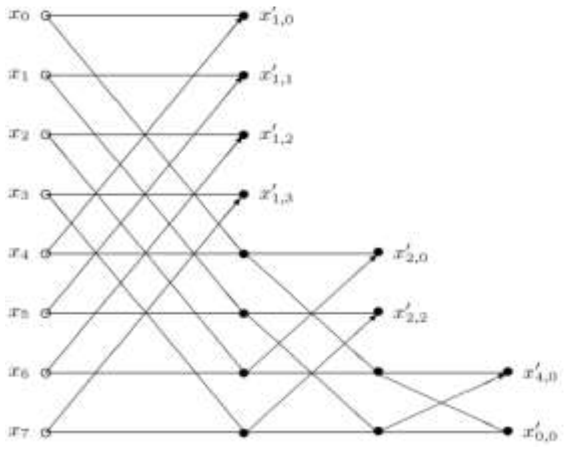


Figure 2. Signal-flow graph of the fast 8-point DPT.

The double numbering of the paired functions refers to the frequency ($p = 2^k$) and time (t). The paired transform represents the discrete-time signal x_n as the unique set of frequency-time signals (or splitting-signals),

$$\left\{\left\{x'_{2^k,0}, x'_{2^k,2^k}, x'_{2^k,2^k2}, \ldots, x'_{2^k,N/2^{k+1}-2^k}\right\}_{k=0:(r-1)}, 1\right\}$$

The components of these splitting-signals are calculated by

$$x'_{2^k,t} = x'_{2^k,t} \circ x_n = \sum_{n=0}^{N-1} x'_{2^k,t}(n) x_n,$$

Where $t = 0:(N/2^{k+1} - 1)$. In the paired transform of the signal, the first $N/2$ components are the splitting-signal with $k = 0$, the next $N/4$ components are the splitting-signal with $k = 1$, and so on. The last component of the transform is $x'_{0,0} = x_1 + x_2 + \cdots + x_{N-1}$. Each splitting-signal defines the N-point DFT of x_n at frequency-points of the corresponding subset

$$T'_p = \{(2m+1)p \bmod N; m = 0:(N/2p - 1)\}.$$

Indeed, the following is valid:

$$X_{\overline{(2m+1)p}} = \sum_{n=0}^{N/(2p)-1} \left(x'_{p,pt} W^{\mathrm{t}}{}_{\mathrm{N/p}}\right) W^{\mathrm{mt}}{}_{\mathrm{N/(2p)}}.$$

The set of N frequency-points $\{0,1,2,\ldots,N-1\}$ is divided by subsets T'_p, where $p = 2^k$, $k = 0:(r-1)$, and $T'_0 = \{0\}$.

Example 1: Consider the signal x_n of length $N = 256$, which is shown in Figure 3 in part a. The paired transform of the signal, which is the set of nine splitting-signals $\left\{x_{T'_1}, x_{T'_2}, x_{T'_4}, x_{T'_8}, \ldots, x_{T'_{128}}, x_{T'_0}\right\}$ is shown in part b. The vertical dashed lines separate the first seven splitting-signals. Figure 4 shows the 256-point DFT X_p of the signal in absolute mode in part a. In part b, the same DFT is shown, but in the order that corresponds to the partition of the frequency-points $\{0,1,2,\ldots,255\}$ by subsets $T'_1, T'_2, T'_4, \ldots, T'_{128}$ and T'_0. The first part with 128 values corresponds to the 128-point DFT of the modified splitting-signal $\left\{x_{T'_1} W^t\right\}$. The next part with 64 values corresponds to the 64-point DFT of the modified splitting-signal $\left\{x_{T'_2} W^t_{128}\right\}$, and so on. Therefore, the following steps are involved in computing the DFT of the input signal x, by using the paired transform:

1) Perform the paired transform $\chi'_N[x]$ over the signal x.
2) Compose $(r+1)$ vectors by dividing the transform into the splitting-signals $\left\{x'_{p,pt}; t = 0:(N/(2p) - 1)\right\}$, where $p = 2^k$, $k = 0:(r-1)$, and $p = 0$.
3) Modify the splitting-signals by the corresponding twiddle factors,

ISSN: 2277-9655
Impact Factor: 4.116

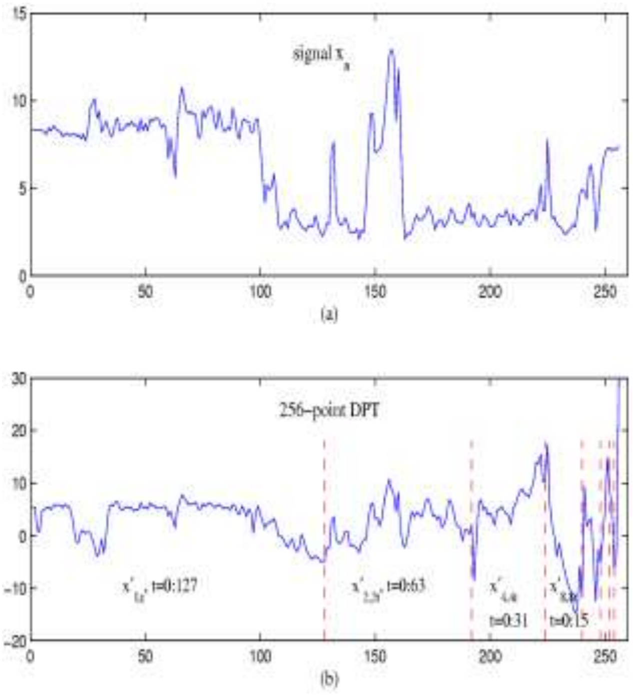


Figure 3. (a) The signal of length 256 and (b) the 256-point discrete paired transform.

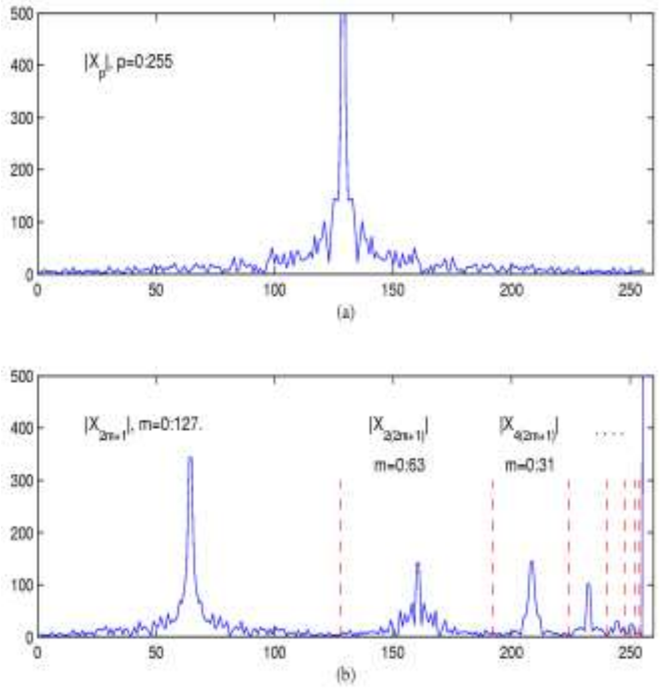


Figure 4. (a) DFT of the signal and (b) DFTs of the modified splitting-signals.

$\{x'_{p,pt}W^t_{N/p}; t = 0{:}\,(N/(2p) - 1)\}$, when $p \neq 0$.

4) Perform the $N/(2p)$-point DFTs over the modified splitting-signals, when $p \neq 0$.
5) Perform the permutation of the output, if needed.

IMPLEMENTATION TECHNIQUES

Implementation on Xilinx FPGAs

Choosing an appropriate tool for FPGA design is of crucial importance as it affects the cost, development time and various other aspects of design. SIMULINK is a platform for multi-domain simulation and Model-Based Design for dynamic systems. It provides an interactive graphical environment and a set of block libraries, and can be extended for different specialized applications. Using SIMULINK one can quickly build up models from libraries of pre-built blocks. For high level design we have chosen Xilinx System Generator. It is a DSP design tool from Xilinx that enables the use of the Mathworks model-based design environment SIMULINK for FPGA design. Xilinx System Generator (XSG) for DSP is a tool which offers block libraries that plugs into SIMULINK tool (containing bit-true and cycle-accurate models of their FPGAs particular math, logic, and DSP functions) [14]-[16]. It is a system-level modeling tool in which designs are captured in the DSP friendly SIMULINK modeling environment using a Xilinx specific blockset. All of the downstream FPGA implementation steps including synthesis and place and route are automatically performed to generate an FPGA programming file. Over 90 DSP building blocks are provided in the Xilinx DSP blockset for SIMULINK. System Generator for DSP, is a system level design tool that is a Blockset for MATLAB SIMULINK. It has different levels of support for the DSP slices.

The hardware modelling of the algorithms is done by using Xilinx's system generator plug-in software tool running under SIMULINK environment provided under the Mathworks's MATLAB software. The functionality of the model is verified using the SIMULINK Simulator and the MODELSIM software as well. The implementation is done using the Xilinx project navigator backend software tools. Many blocks of custom MATLAB code (.m files) were however needed for the design and the hardware generated for these blocks was not optimized. We used DSP IP with DSP slices support, and the DSP slice to build custom functions, the DSP slice macro to simplify sequential instructions.

We have implemented various architectures for radix-2 and paired transform processors on Xilinx Virtex-II Pro, Virtex-4, Virtex-5 FPGAs. As there are embedded dedicated multipliers and embedded block RAMs available, we can use them without using distributed logic, which economize some of the CLBs. As we are having Extreme DSP slices [9] on Virtex-4 FPGAs, DSP48E Slices [7] on Virtex-5 FPGAs we have utilized them to improve speed performance of these 2 FFTs and to compare their speed performances. As most of the transforms are applied on complex data, the arithmetic unit always needs two data points at a time for each operand (real part and complex part), dual port RAMs are very useful in all these implementation techniques.

In the Fast Fourier Transform process the butterfly operation is the main unit on which the speed of the whole process of the FFT depends. So the faster the butterfly operation, the faster the FFT process. The adders and subtractors are implemented using the LUTs (distributed arithmetic). The inputs and outputs of all the arithmetic units can be registered or non-registered.

We have considered the implementation of both embedded and distributed multipliers; the latter are implemented using the LUTs in the CLBs. The three considerations for inputs/outputs are with non-registered inputs and outputs, with registered inputs or outputs, and with registered inputs and outputs. To implement butterfly operation for its speed improvement and resource requirement, we have implemented both multiplication procedures basing on the availability of number of embedded multipliers (especially for Virtex-II Pro FPGAs), and design feasibility; and design feasibility using DSP Slices.

The various architectures proposed for implementing radix-2 and paired transform processors are single memory (pair) architecture, dual memory (pair) architecture and multiple memory (pair) architectures. We applied the following two best butterfly techniques for the implementation of the processors on the FPGAs.

1. One with Distributed multipliers, and DSP slices with fully pipelined stages. (Best in case of performance)
2. One with embedded multipliers and DSP slices and one level pipelining. (Best in case of resource utilization)

Single memory (pair) architecture (shown in Figure 5) is suitable for single snapshot applications, where samples are acquired and processed thereafter. The processing time is typically greater than the acquisition time. The main disadvantage in this architecture is while doing the transform process we cannot load the next coming data. We have to wait until the current data is processed. So we proposed dual memory (pair) architecture for faster sampling rate applications (shown in Figure 6). In this architecture there are three main processes for the transformation of the sampled data. Loading the sampled data into the memories, processing the loaded data, reading out the processed data. As there are two pairs of dual port memories available, one pair can be used for loading the incoming sampled data, while at the same time the other pair can be used for processing the previously loaded sampled data.

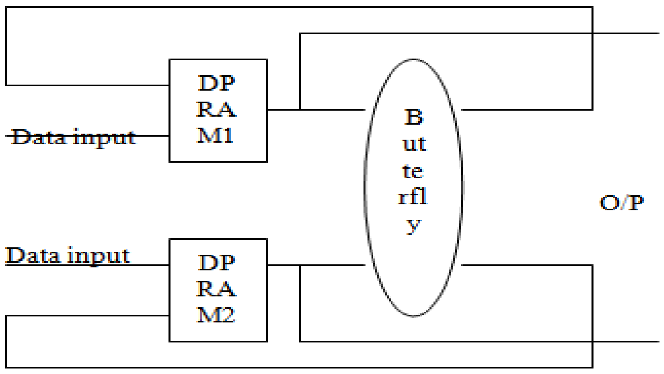


Figure 5. Single memory (pair) architecture

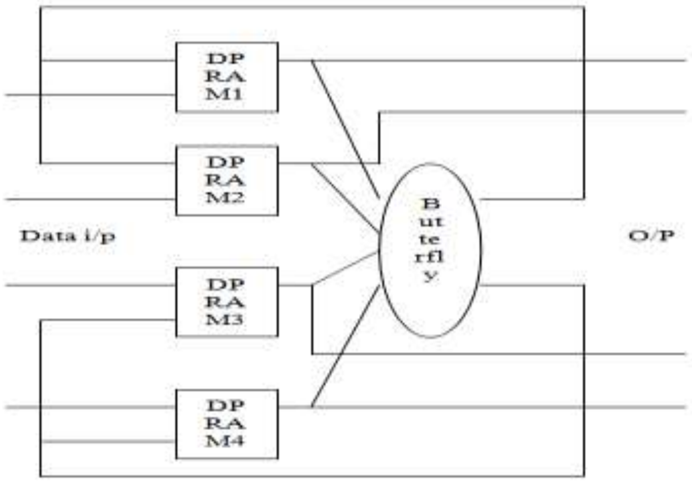


Figure 6. Dual memory (pair) architecture

For further sampling rate improvements we proposed multiple memory (pair) architecture (shown in Figure 7). This is the best of all architectures in case of very high sampling rate applications, but in case of hardware utilization it uses lot more resources than any other architecture. In this model there is a memory set, one arithmetic unit for each iteration. The advantage of this model over the previous models is that we do not need to wait until the end of all iterations (i.e. whole FFT process), to take the next set of samples to get the FFT process to be started again. We just need to wait until the end of the first iteration and then load the memory with the next set of samples and start the process again. After the first iteration the processed data is transferred to the next set of RAMs, so the previous set of RAMs can be loaded with the next coming new data samples. This leads to the increased sampling rate.

Coming to the implementation of the paired transform based DFT algorithm, there is no complete butterfly operation, as that in case of radix-2 algorithm. According to the mathematical description given in the Section 2, the arithmetic unit is divided into two parts, addition part and multiplication part. This makes the main difference between the two algorithms, which causes the process of the DFT completes earlier than the radix-2 algorithm. The addition part of the algorithm for 8-point transform is shown in Figure 2.

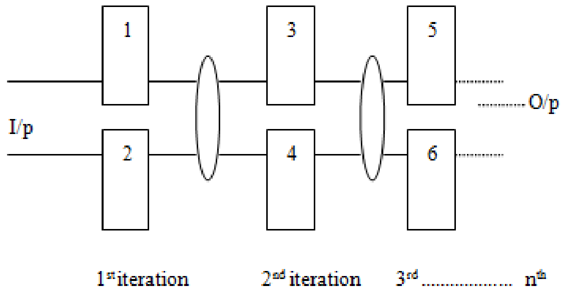
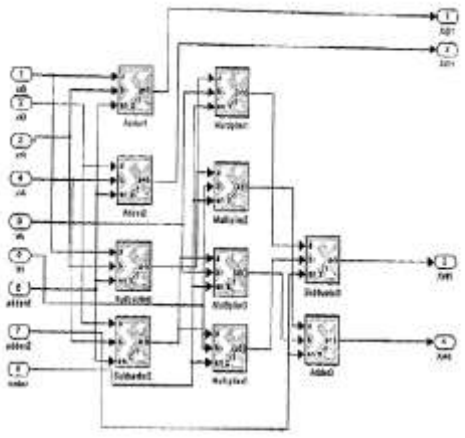
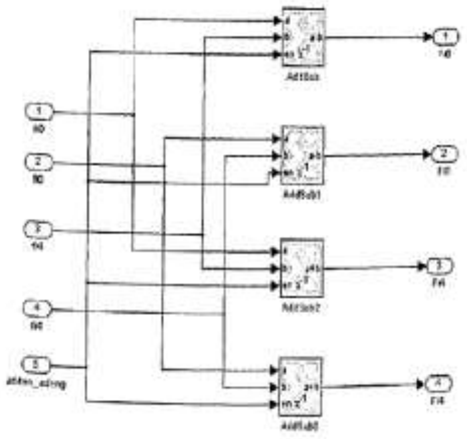


Figure 7. Multiple memory (pair) architecture
(Transform length = N = 2^n)
(1,2);(3,4);(5,6) ---- (-,-) memory pairs for each iteration.
----- Butterfly unit for each iteration.

The SIMULINK model diagrams for butterfly operation for both FFTs are given in Figure 8.



(a)



(b)
Figure 8. SIMULINK models (a) for butterfly diagram for N=8 Cooley-Tukey FFT (b) for the addition part shown in figure 2 of Paired transform based FFT: Grigorayn FFT.

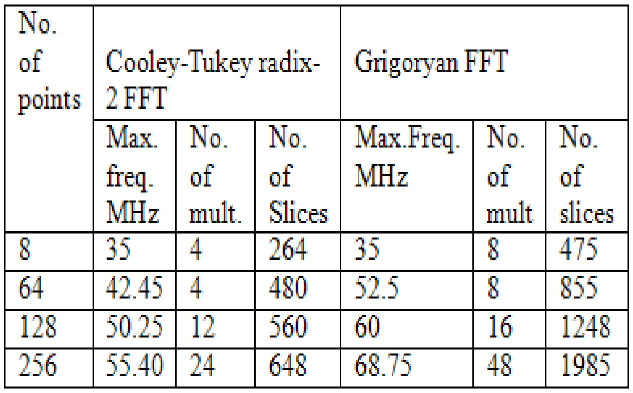
The architectures are implemented for the 8-point, 64-point, 128-point and 256-point transforms for Xilinx Virtex-II Pro, Virtex-4 and Virtex-5 FPGAs. The radix-2 FFT algorithm is efficient in case of resource utilization and the paired transform algorithm is very efficient in case of speed of operation and hence higher sampling rate applications.

Implementation on Texas Instruments DSP Processors.
We have developed Programs for the Cooley-Tukey FFT and Grigoryan FFT algorithms in C language specific for the Texas Instruments DSP processors by using the software Texas Instruments Code Composer Studio Integrated development environment. The C programming is done for the implementation on TMS fixed point DSP Processors: TMS320C6748, TMS320C5416, and TMS320C5515. The features of DSP specific capabilities, advanced break points, the conditional or hardware break points for C expressions, local variable and registers are utilized to the best for simple and efficient implementations. The advanced memory window is used for the observation of memory at each level. Using the CC studio we can be able to easily and quickly measure code performance and ensure the efficient use of the DSP target's resources during debug and development sessions. Detailed simulations are observed using the features of the CC Studio for finding the number of CCs and finding the speed of operation of the two algorithms while implementing on the three DSP processors.

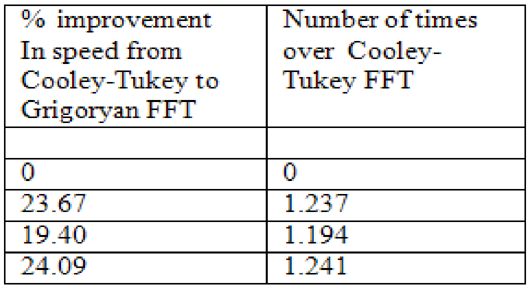
PRIMILIMINARY IMPLEMENTATION RESULTS

Results on Xilinx FPGAs
We have implemented all 3 different memory pair architectures explained in Section 3. But here we are providing the results for multiple memory pair architecture in Figure 7 as it is the most efficient of all. We are showing the results and efficiency comparison of both Cooley-Tukey and Grigoryan FFT on Xilinx Virtex-II Pro, Virtex-4 and Virtex-5 FPGAs for the same architectures. Then we are showing the % improvement of the Grigoryan FFT over the Cooley-Tukey FFT and providing how many times the Grigoryan FFT is faster than the Cooley-Tukey FFT. Then we are comparing the three FPGA platforms to verify which one is better platform for implementing our architectures. From the results we can easily identify that the Grigoryan FFT is much faster than the Cooley-Tukey FFT.

Tables 1,2,3 and 4 show the implementation results of the two algorithms on all the considered Xilinx FPGAs. From these results we can see that the Grigoryan FFT is always faster than the Cooley-Tukey FFT algorithm. Thus paired-transform based algorithm can be used for higher sampling rate applications. In military applications, while doing the process, only some of the DFT coefficients are needed at a time. For this type of applications paired transform can be used as it generates some of the coefficients earlier, and also it is very fast. As of verification of better platform on which we can implement our architectures it is found that Virtex-5 FPGAs are better platform for our architectural implementations. By observing the results the Grigoryan FFT is performing at most 24.09 % faster. If we observe the number of times speed, it is clear that the Grigoryan FFT is working at most 1.358 times faster than the Cooley-Tukey FFT.



No. of points	Cooley-Tukey radix-2 FFT			Grigoryan FFT		
	Max. freq. MHz	No. of mult.	No. of Slices	Max.Freq. MHz	No. of mult	No. of slices
8	35	4	264	35	8	475
64	42.45	4	480	52.5	8	855
128	50.25	12	560	60	16	1248
256	55.40	24	648	68.75	48	1985



% improvement In speed from Cooley-Tukey to Grigoryan FFT	Number of times over Cooley-Tukey FFT
0	0
23.67	1.237
19.40	1.194
24.09	1.241

Table 1. Efficient performance of the Grigoryan FFT over Cooley-Tukey FFT, on Xilinx Virtex-II Pro FPGAs. Table showing the sampling rates and the resource utilization summaries for both the algorithms, implemented on the Virtex-II Pro FPGAs.

ISSN: 2277-9655
Impact Factor: 4.116

No. of points	Cooley-Tukey radix-2 FFT			Grigoryan FFT		
	Max. freq. MHz	No. of Slices	No. of DSP48E slices	Max. freq MHz	No. of Slices	No.of DSP48 E slices
8	48	200	4	52	350	4
64	55.25	375	6	60	700	8
128	60.50	450	11	80	1000	26
256	75	560	14	85	1645	82

% improvement In speed from Cooley-Tukey to Grigoryan FFT	Number of times over Cooley-Tukey FFT
8.33	1.083
8.59	1.086
32.23	1.322
13.33	1.133

Table 2. Efficient performance of the Grigoryan FFT over the Cooley-Tukey FFT, on Virtex-5 FPGAs. Table showing the sampling rates and the resource utilization summaries for both the algorithms, implemented on the Virtex-5 FPGAs. We have utilized DSP48E slices in this, which is making us much faster than Virtex-II Pro FPGAs.

No. of points	Cooley-Tukey radix-2 FFT			Grigoryan FFT		
	Max. freq. MHz	No. of Slic-es	No. of Extreme DSP slices	Max. freq MHz	No. of Slices	No.of Extreme DSP slices
8	41.38	258	4	50	450	6
64	48	400	7	56.09	800	8
128	53	496	12	72	1120	28
256	67	598	20	79.95	1735	86

% improvement In speed from Cooley-Tukey to Grigoryan FFT	Number of times over Cooley-Tukey FFT
20.83	1.208
16.85	1.169
35.85	1.358
19.33	1.193

Table 3. Efficient performance of the Grigoryan FFT over the Cooley-Tukey FFT, on Virtex-4 FPGAs. Table showing the sampling rates and the resource utilization summaries for both the algorithms, implemented on the Virtex-4 FPGAs. We have utilized Extreme DSP slices in this, which is making us much faster than Virtex-II Pro FPGAs.

Number Of Sample points of FFT	% Improvement in speed over Xilinx Virtex-II Pro FPGAs			
	Cooley-Tukey FFT		Grigoryan FFT	
	Virtex - 5	Virtex - 4	Virtex - 5	Virtex - 4
8	37.14	18.23	48.57	42.86
64	30.15	13.07	14.29	6.84
128	20.40	5.47	33.33	20
256	35.38	20.94	23.64	16.29

Table 4. The percentage improvement in speed of operation over Virtex-II Pro FPGAs, of Virtex-5 and Virtex-4; of both Cooley-Tukey and Grigoryan FFT algorithms. It shows clearly that Virtex-5 plat form is better than Virtex-4 and also Virtex-II pro, in terms of speed of operation.

Results on Texas Instruments DSP processors

After implementing on the three TMS DSP processors we have observed the performances for N = 16, 32,64, 128,256,512,1024 point FFTs for both Cooley-Tukey FFT and Grigoryan FFT. The implementation results are given in Tables 5-10 for all the three DSP processors. By observing on all the three TMS DSP processors, the % speed improvement of Grigoryan FFT over Cooley-Tukey FFT it is clear that the Grigoryan FFT is performing much better and can be utilized for higher sampling rates of operation. The Grigoryan FFT is performing at most 70% faster. If we observe the number of times speed, it is clear that the Grigoryan FFT is working at most 1.7 times faster than the Cooley-Tukey FFT. An example graphical representation, for TMS320C6748 DSP, of the speed improvement of the Grigoryan FFT over the Cooley-Tukey FFT is given in the Figure 9.

No. of Samples	Radix-2 FFT	Paired transform based FFT	%improvement
16	100630	85601	18
32	274613	217498	26
64	388029	294799	32
128	1530693	1134670	35
256	1750023	1578720	48
512	4762375	2983408	60
1024	15123833	8987561	68

Table 5 Performance comparison of the two algorithms on DSP processor TMS320C6748 (fixed point) processor.

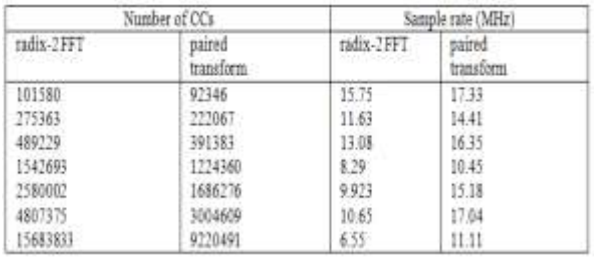
Number of CCs		Sample rate (MHz)	
radix-2 FFT	paired transform	radix-2 FFT	paired transform
100630	85601	19.75	21.76
274613	217498	12.63	18.41
488029	294799	15.08	19.35
1530693	1134670	10.29	14.54
1750023	1578720	12.923	18.81
4762375	2983408	13.56	19.40
15123833	8987561	10.45	14.44

Sample rate (MHz)		Number of times over Cooley-Tukey FFT
radix-2 FFT	paired transform	
19.75	21.76	1.102
12.63	18.41	1.458
15.08	19.35	1.283
10.29	14.54	1.413
12.923	18.81	1.456
13.56	19.40	1.431
10.45	14.44	1.382

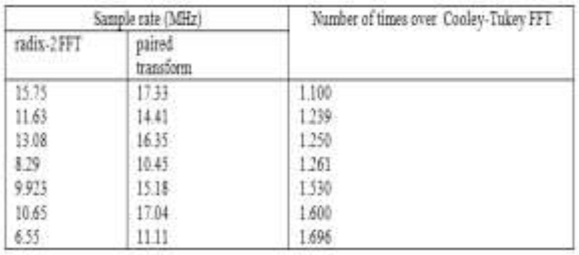
Table 6 Table showing the sampling rate of both the algorithms (starting form N = 16 to N = 1024) for TMS320C6748.

No. of Samples	Radix-2 FFT	Paired transform based FFT	%improvement
16	101580	92346	10
32	275363	222067	24
64	489229	391383	25
128	1542693	1224360	26
256	2580026	1686276	53
512	4807375	3004609	60
1024	15683833	9220491	70

Table 7 Performance comparison of the two algorithms on DSP processor TMS320C5416 (fixed point) processor.



Number of CCs		Sample rate (MHz)	
radix-2 FFT	paired transform	radix-2 FFT	paired transform
101580	92346	15.75	17.33
275363	222067	11.63	14.41
489229	391383	13.08	16.35
1542693	1224360	8.29	10.45
2580002	1686276	9.923	15.18
4807375	3004609	10.65	17.04
15683833	9220491	6.55	11.11



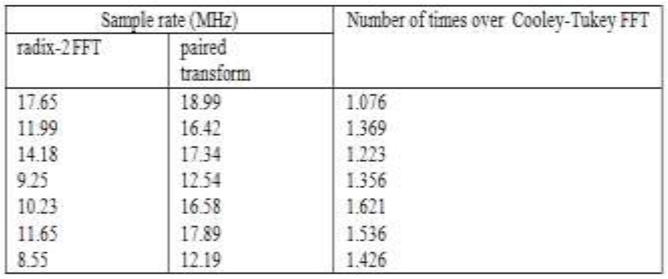
Sample rate (MHz)		Number of times over Cooley-Tukey FFT
radix-2 FFT	paired transform	
15.75	17.33	1.100
11.63	14.41	1.239
13.08	16.35	1.250
8.29	10.45	1.261
9.923	15.18	1.530
10.65	17.04	1.600
6.55	11.11	1.696

Table 8 Table showing the sampling rate of both the algorithms (starting form N = 16 to N = 1024) for TMS320C5416.

No. of Samples	Radix-2 FFT	Paired transform based FFT	%improvement
16	101080	90235	12
32	275160	220090	25.02
64	461111	330273	39.61
128	1540680	1201321	28.25
256	2280014	1609342	41.67
512	4803459	3004099	59.89
1024	15230721	9002381	69.18

Table 9 Performance comparison of the two algorithms on DSP processor TMS320C5515 (fixed point) processor.

Number of CCs		Sample rate (MHz)	
radix-2 FFT	paired transform	radix-2 FFT	paired transform
101080	90235	17.65	18.99
275160	220090	11.99	16.42
461111	330273	14.18	17.34
1540680	1201321	9.25	12.54
2280014	1609342	10.23	16.58
4803459	3004099	11.65	17.89
15230721	9002381	8.55	12.19



Sample rate (MHz)		Number of times over Cooley-Tukey FFT
radix-2 FFT	paired transform	
17.65	18.99	1.076
11.99	16.42	1.369
14.18	17.34	1.223
9.25	12.54	1.356
10.23	16.58	1.621
11.65	17.89	1.536
8.55	12.19	1.426

Table 10 Table showing the sampling rate of both the algorithms (starting form N = 16 to N = 1024) for TMS320C5515.

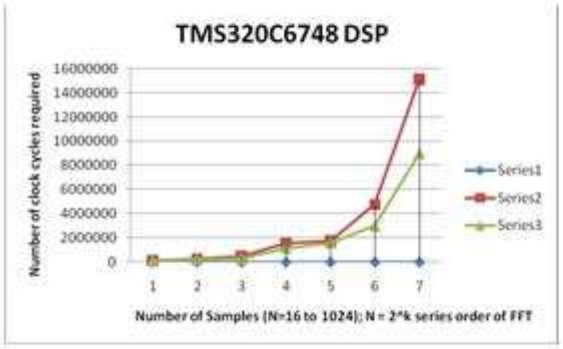


Figure 9. For TMS320C6748 DSP. On Y-axis the number of clock cycles taken to process the FFT of size N = 16,32,64,128,256,512,1024. N on X-axis. Series 2,3 are for the number of clock cycles taken for the Cooley-Tukey FFT and Grigoryan FFT respectively, it is clear that the Grigoryan FFT is going faster than the Cooley-Tukey FFT.

CONCLUSION AND FURTHER RESEARCH

In this paper we have shown that with our FFT processors architectures on Xilinx FPGAs and TMS DSPs the paired transform based Grigoryan FFT algorithm is faster and can be used at higher sampling rates than the Cooley-Tukey FFT at an expense of high resource utilization. It is observed that implementations for the Grigoryan FFT we are the first to design and we are the first and may be the best implementers using SIMULINK. After studying the solution of implementation method using SIMULINK we demonstrated its hardware feasibility and visual interface through Xilinx system generator. We have explored the feasibility of CC studio for TMS DSP processors for the Grigoryan FFT. We have proved that on Xilinx FPGAs and TMS DSPs, Grigoryan FFT is performing at most 1.358 and 1.7 times faster than the Cooley-Tukey FFT respectively. Which is a good improvement in speed over the Cooley-Tukey FFT. As a technological confirmation, for our implementations especially for the Grigoryan FFT (FFT of interest) out of the three FPGA platforms we conclude that Virtex-5 FPGAs is found to be better platform.

As a further research, these architectures are extendible with even best utilization of the features of DSP slices on FPGAs, by efficiently exploiting parallelism of FPGA; and explicit utilization of MAC engines on DSP processors, which can lead to most efficient implementations and the Grigoryan FFT can show even much better performance.

ACKNOWLEDGEMENTS

This is to acknowledge Prof. Dr. Parimal A. Patel, Prof. Dr. Artyom M. Grigoryan of The University of Texas at San Antonio, & Bindu Tushara D, B.Tech double gold medallist: 2009 JNTU-H. Especially Both Profs for their valuable guidance for the first part of this research series as Mr. Narayanam’s Master’s Thesis, which is indexed in Google books. Mr. Narayanam also would like to acknowledge University of Ottawa research facility providers for providing essential tools while he was a biomedical engineering research student under Prof. Dr. Hilmi Dajani.

REFERENCES

[1] J.W. Cooley and J.W. Tukey, "An algorithm the machine computation of complex Fourier series," Math. Comput. 1965. vol. 9, no. 2, pp. 297-301.
[2] A.M. Grigoryan, and S.S. Agaian, ``Split manageable efficient algorithm for Fourier and Hadamard transforms," IEEE Trans. on Signal Processing, vol. 48, no. 1, pp. 172-183, Jan. 2000.
[3] A.M. Grigoryan,``2-D and 1-D multi-paired transforms: Frequency-time type wavelets," IEEE Trans. on Signal Processing}, vol. 49, no. 2, pp. 344-353, Feb. 2001.
[4] A. M. Grigoryan and S. S. Agaian, *Multidimensional Discrete Unitary Transforms: Representation, Partitioning, and Algorithms*, New York: Marcel Dekker, 2003.
[5] A.M. Grigoryan and M.M. Grigoryan, *Brief Notes in Advanced DSP: Fourier Analysis With MATLAB*, CRC Press Taylor and Francis Group, 2009.
[6] Virtex-II Pro platform FPGAs: detailed description http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf

[7] Virtex-5 platform FPGAs: detailed description http://www.xilinx.com/support/documentation/virtex-5_user_guides.htm
[8] N. Ranganadh, P. Patel, and A.M. Grigoryan, ``Case study of Grigoryan FFT onto FPGAs and DSPs," IEEE Proceedings, IEEE ICECT 2012. 4th International Conference on Electronics Computer Technology, Kanya Kumari – 2012.
[9] Virtex-4 platform FPGAs: detailed description http://www.xilinx.com/support/documentation/virtex-4_user_guides.htm
[10] The scientist and engineer's guide to Digital Signal Processing, Dr. Steven W. Smith.
[11] N. Ranganadh, P. Patel, and A.M. Grigoryan, ``Implementation of the DFT using radix-2 and paired transform algorithms," 17th International Conference on Computer Applications in Industry and Engineering, CAINE-2004, Orlando, Florida USA, Nov. 17-19, 2004.
[12] N. Ranganadh, P. Patel, and A.M. Grigoryan, ``Performances of Texas instruments DSP and Xilinx FPGAs for Cooley-Tukey and Grigoryan FFT algorithms", Wolters Kluwer and Medknow International Journal of Engineering and Technology, Jul-Dec 2011, vol 1, Issue 2.
[13] CC Studio detailed description: http://processors.wiki.ti.com/index.php/Code_Composer_Studio_v5_Users_Guide
[14] Systems generator for DSP applications on FPGA: http://www.academia.edu/2475772/System_Generator_The_State-of-art_FPGA_Design_tool_for_DSP_Applications
[15] FPGA:DSP functions in SIMULINK http://www.mathworks.in/company/newsletters/articles/dsp-functions-on-fpgas.html
[16] FPGAs Vs. ASICS http://www.xilinx.com/fpga/asic.htm
[17] TMS320C5416, TMS320C6748, TMS320C5515 manuals:www.ti.com
[18] N. Ranganadh, P.A. Patel, and A.M. Grigoryan, "Case Study of Grigoryan FFT onto FPGAs and DSPs," *IJFCC - International Journal of Future Computer and Communication,"* vol. 2, no. 6, pp. 678-681. December 2013
[19] Narayanam Ranganadh and Rekha Andal Vangala, "A TMS DSP processor based case study of Grigoryan FFT performance over Cooley-Tukey FFT (TMS320C6748, TMS320C5515)," *IOSR Journal of Engineering (IOSRJEN),* vol 3, no 1, pp. 55-60, January 2013
[20] Narayanam Ranganadh and Nageswara rao Dhanavath, "A TMS DSP processor based case study of Grigoryan FFT performance over Cooley-Tukey FFT (TMS320C5416, TMS320C5515)," *Asian Journal of Curernt Engineering and Maths,* vol 2, no 1, pp. 50-52, January-February 2013