Case Study of Grigoryan FFT onto FPGAs and DSPs

Narayanam Ranganadh, Parimal A Patel, and Artyom M. Grigoryan

***Abstract*—Frequency analysis plays vital role in the applications like cryptanalysis, steganalysis, system identification, controller tuning, speech recognition, noise filters, etc. Discrete Fourier Transform (DFT) is a principal mathematical method for the frequency analysis. 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 TMS320C62x DSP and also on the Virtex-II pro FPGAs. Finally we show that the paired-transform based algorithm of the FFT is faster than the radix-2 FFT, consequently it is useful for higher sampling rates.**

***Index Terms*—Frequency analysis, fast algorithms, DFT, FFT, paired transforms.**

I. Introduction

In the past decade, fast orthogonal transforms have been widely used in areas of data compression, pattern recognition and image reconstruction, interpolation, linear filtering, and spectral analysis. The suitability of unitary transforms in each of the above applications depends on the properties of their basis functions as well as on the existence of fast algorithms, including parallel ones. 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 [5]. 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 splittings 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. 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 [5]. 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 this paper we present the implementation techniques and their results for two different fast DFT algorithms. The difference between the algorithm developments lies in the way the two algorithms use the splitting of the DFT. The two fast algorithms considered are radix-2 and paired transform [2] algorithms. The implementation of the algorithms is done both on the TMS320C62x digital signal processor and also on the Xilinx Viretx-II FPGAs. The performance of the two algorithms is compared in terms of their sampling rates and also in terms of their hardware resource utilization.

Section II presents the paired transform decomposition used in paired transform in the development of Grigoryan FFT. Section III presents the implementation techniques for the radix-2 and paired transform algorithms on FPGAs. Section IV presents the results. Finally with the Section V we conclude the work and put forward some suggestions for further sampling rate improvements.

II. Decomposition Algorithm of the Fast DFT using Paired Transform

In this algorithm the decomposition of the DFT is done by using the *paired transform [2]*. Let $\{x(n)\}$, n = 0:(N-1) be an input signal, N>1. Then the DFT of the input sequence $x(n)$ is

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \text{ , k =0:(N-1)} \quad (1)$$

which is in matrix form

$$\mathrm{X} = [F_N]x \quad (2)$$

where $X(k)$ and $x(n)$ are column vectors, the matrix $F_N = \left\| W_N^{nk} \right\|_{n,k=(0:N-1)}$, is a permutation of X.

$$F_N = diag\{[F_{N1}],[F_{N2}],.....[F_{Nk}]\}[\overline{W}][\chi_N^{'}] \quad (3)$$

Which shows the applying transform is decomposed into short transforms F_{Ni}, i = 1: k. Let S_F be the domain of the transform F the set of sequences f over which F is defined. Let (D; σ) be a class of unitary transforms revealed by a partition σ. For any transform $F \in$ (D; σ), the computation is performed by using paired transform in this particular algorithm. To denote this type of transform, we introduce *"paired functions [2]."*

Let $p, t \in$ period N, and let

Manuscript received December 19, 2012; revised February 23, 2013.

Narayanam Ranganadh is with the Department of Electrical Engineering, The University of Texas at San Antonio (e-mail: ranganadh.narayanam@gmail.com).

DOI: 10.7763/IJFCC.2013.V2.251

$$\chi_{p,t}(n) = \begin{cases} 1; np = t(\text{mod } N); \\ 0; otherwise. \end{cases} \quad \text{n = 0: (N-1)} \qquad (4)$$

Let L be a non trivial factor of the number N, and $W_L = e^{2\Pi / L}$, then the complex function

$$\chi'_{p,t} = \chi'_{p,t;L} = \sum_{k=0}^{L-1} W_L^k \chi_{p,t+kN/l} \qquad (5)$$

$T = 0: (N/L - 1)$, p€ to the period 0: N-1

Is called L-paired function [2]. Basing on these paired functions the complete system of paired functions can be constructed. The totality of the paired functions in the case of $N=2^r$ is

$$\{\{\chi_{2^n,2^n t}; t = 0 : (2^{r-n-1} - 1), n = 0 : (r-1)\}, 1\} \qquad (6)$$

Now considering the case of $N = 2^r N_1$, where N_1 is odd, $r \geq 1$ for the application of the paired transform.

- The totality of the partitions is
$$\sigma' = (T'_{1;2}, T'_{2;2}, T'_{4;2}, ..., T'_{2^r;2})$$
- The splitting of F_N by the partition σ' is
$$\{F_{N/2}, F_{N/4}, ..., F_{N/2^r}, F_{N1}\}$$
- The matrix of the transform can be represented as
$$[F_N] = \left(\oplus_{n=1}^{r} [F_{Ln}] \right) [\overline{W}][\chi'_n]$$
$$L_n = N/2^{n+1}, L_r = N_1.$$

where the $[\overline{W}]$ is diagonal matrix of the twiddle factors. The steps involved in finding the DFT using the paired transform are given below:

1) Perform the L-paired transform g = $\chi'_N(x)$ over the input x
2) Compose r vectors by dividing the set of outputs so that the first L^{r-1} elements $g_1, g_2, ..., g_{L^{r-1}}$ compose the first vector X_1, the next L^{r-2} elements $g_{Lr-1}+1, ..., g_{Lr-1+Lr-2}$ compose the second vector X_2, etc.
3) Calculate the new vectors Y_k, k=1:(r-1) by multiplying element-wise the vectors X_k by the corresponding turned factors1
4) $, W_t, W_t^2, ..., W_t^{t/L-1}, (t = L^{r-k}), (t=Lr-k)$. Take $Y_r = X_r$
5) Perform the L_{r-k}-point DFT's over Y_k, k=1: r
6) Make the permutation of outputs, if needed.

III. Implementation Techniques

We have implemented various architectures for radix-2 and paired transform processors on Virtex-II pro FPGAs [3]. As there are embedded multipliers [3] and embedded block RAMs [3] available, we can use them without using distributed logic, which economize some of the CLBs [3]. 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), dualport 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.

Various possible implementations of multipliers one can consider are:

Embedded multiplier:

- With non-registered inputs and outputs
- With registered inputs or outputs, and
- With registered inputs and outputs.

Distributed multiplier: Distributed multipliers are implemented using the LUTs in the CLBs. These can also be implemented with the above three possible ways. Various considerations made to implement butterfly operation for its speed improvement and resource requirements. The results of these techniques are tabulated in Table I.

TABLE I: The Results of Various Butterfly Implementations.

	Embedded multipliers	Pipelining (Latency)	Ip/Op registers	No.of Slices	No.of Muls	Freq (MHz)
1.	Yes	Yes(1)	No	128	4	42.31
2.	Yes	No	No	128	4	26.84
3.	No	No	No	734	0	26.45
4.	Yes	Yes	Yes	288	4	108.16
5.	Yes	No	Yes	148	4	48.71
6.	No	Yes(1)	Yes	760	0	111.50
7.	No	No	Yes	736	0	38.19

By observing the results we can say that the butterfly operation with pipelined multipliers (pipelined to maximum extent possible (4)), distributed arithmetic operations, and registering all the inputs and outputs of the all arithmetic units provides the fastest butterfly operation. 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 Virtex-II pro FPGAs [3].

One with Distributed multipliers, with fully pipelined stages. (Best in case of performance)

One with embedded multipliers and one level pipelining. (Best in case of resource utilization)

Single memory (pair) architecture (shown in Fig. 1) 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 Fig. 2). 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. For further sampling rate improvements we proposed multiple memory (pair) architecture (shown in Fig. 3). 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.

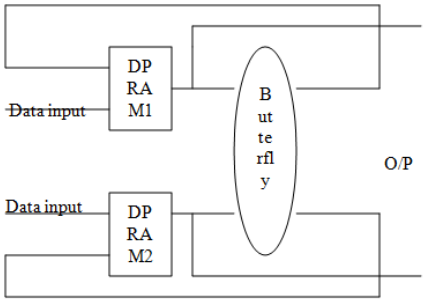

Fig. 1. Single memory (pair) architecture

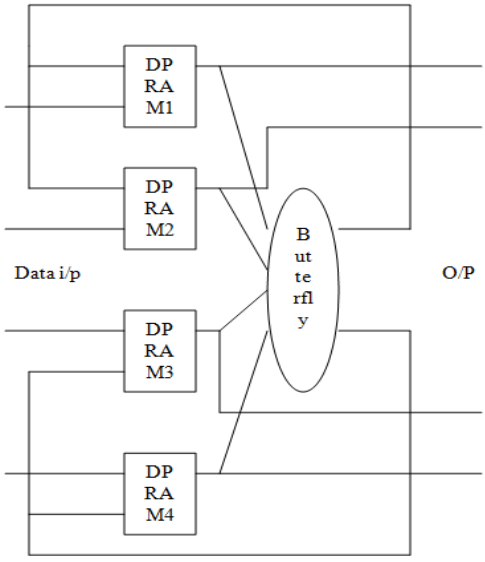

Fig. 2. Dual memory (pair) architecture

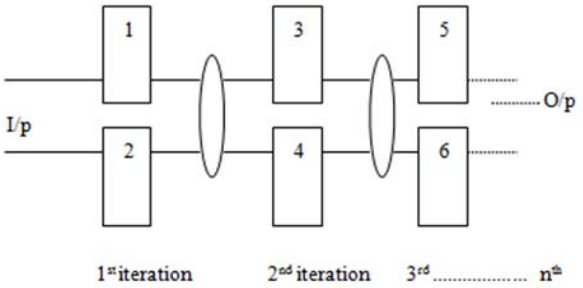

Fig. 3. Multiple memory (pair) architecture
(Transform length = N = 2^n)
(1,2);(3,4);(5,6) ---- (-,-) memory pairs for each iteration.

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 II, 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 Fig 4. The architectures are implemented for the 8-point and 64-point transforms. The radix-2 FFT algorithm is efficient in case of resource utilization and the paired transform algorithm is very efficient in case of higher sampling rate applications.

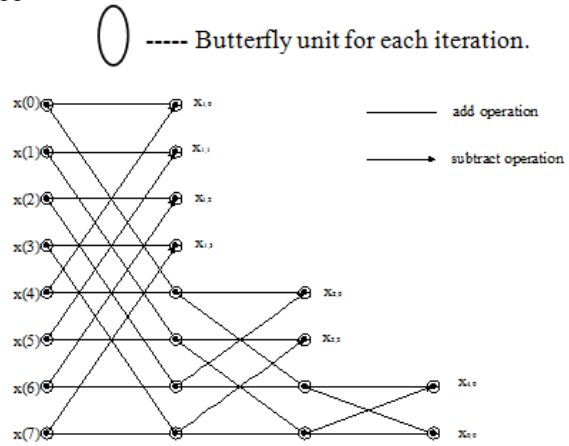

Fig. 4. Figure showing the addition part of the 8-point paired transform based DFT

IV. The Implementation Results

Results obtained on TMS320C62x digital signal processor: The software used for implementing on the DSP is Texas instruments code composer studio [4]. We used C programming language for implementation. TABLE II and Table III show the implementation results and the comparison between the two algorithms on the DSP processor.

Table II shows that the paired algorithm of DFT is much faster than the radix-2 algorithm. Going to higher transform lengths, paired algorithm gives the higher percentage improvement over the radix-2 algorithm. TABLE III shows the time required and the sampling rates that the two algorithms can be operated at.

TABLE II: Performance comparision of the two algorithms on DSP processor.

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	258002	1686276	53
512	4807375	3004609	60
1024	15683833	9220491	70

Results obtained on Virtex-II pro FPGAs: The hardware modeling 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.

TABLE III: TABLE SHOWING THE SAMPLING RATE OF BOTH THE ALGORITHMS (STARTING FROM N = 8 TO *N* = 1024)

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

TABLE IV: TABLE SHOWING THE SAMPLING RATES AND THE RESOURCE UTILIZATION SUMMARIES FOR BOTH THE ALGORITHMS, IMPLEMENTED ON THE VIRTEX-II PRO FPGAS

No. of points	Max. freq (radix-2 alg.)	No. of mult (radix-2 alg.)	No. of slices (radix-2 alg.)	Max. freq (paired alg.)	No. of mult (paired alg.)	No. of slices (paired alg.)
8	35 MHZ	4	264	35 MHz	8	475
64	42.45 MHz	4	480	52.5 MHz	8	855

Table IV shows the implementation results of the two algorithms on the Virtex-II pro FPGAs. From TABLE II, III, IV we can see that paired transform is always faster than the radix-2 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.

V. CONCLUSION

In this paper we have shown that both on DSPs and also on FPGAs the paired transform based algorithm is faster and can be used at higher sampling rates than the radix-2 FFT at an expense of high resource utilization.

1) In all implementations on FPGAs, the number of bits used for the data is 16-bits. So all the multipliers here are used as 16-bit multipliers. The size of the multipliers used was 18-bit multipliers. For instance, if there are some applications using only 8-bit data, then one can use the 40 dedicated multipliers as 80 multipliers, as two multiplications can be implemented by using a single embedded multiplier as long as the sum of the two products bits is less than 36 bits.

2) In the implementations on the DSP if the MAC engines are used explicitly, then there may be a possibility of better comparison between the algorithms. Also one can see some more speed improvement in the DFT processes.

REFERENCES

[1] J.W. Cooley and J.W. Tukey, "An algorithm for the machine calculation of complex Fourier Series," *Math. comput.* vol. 19, pp. 297-301, 1965.

[2] A. M. Grigoryan and S. S. Agaian, "Split Manageable Efficient Algorithm for Fourier and Hadamard transforms," *Signal Processing, IEEE Transactions on*, vol. 48, no. 1, pp. 172 – 183, Jan.2000.

[3] Virtex-II pro platform FPGAs: detailed description. [Online]. Available: http://direct.xilinx.com/bvdocs/publications/ds031-2.pdf

[4] *CC Studio getting started guide*, TMS320C62x.

[5] S. W. Smith, *The scientist and engineer's guide to Digital Signal Processing*, California Technical Publishing.

Ranganadh Narayanam is an assistant professor in the department of Electronics & Communications Engineering in Bharat Institute of Engineering & Technology (BIET). This research is continuation of the research done in the Univeristy of Texas at San Antonio under the guidance of Dr. Parimal A. Patel, Dr. Artyom M. Grigoryan, as my master's thesis. Mr.Narayanam, a research student in the area of "Brain Stem Speech Evoked Potentials" under the guidance of Dr. Hilmi Dajani of University of Ottawa,Canada. He was also a research student in The University of Texas at San Antonio under Dr. Parimal A Patel,Dr. Artyom M. Grigoryan, Dr Sos Again, Dr. CJ Qian,. in the areas of signal processing and digital systems, control systems. He worked in the area of Brian Imaging in University of California Berkeley. Mr. Narayanam has done some advanced learning in the areas of DNA computing, String theory and Unification of forces, Faster than the speed of light theory with worldwide reputed persons and world's top ranked universities. Mr. Narayanam's research interests include neurological Signal & Image processing, DSP software & Hardware design and implementations, neurotechnologies. Mr. Narayanam can also be contacted at rnara100@gmail.com, rnara100@biet.ac.in, ranganadh.narayanam@gmail.com, rnara100@uottawa.ca

Parimal A. Patel did his doctoral studies in UT Austin. He was a professor and Chair of the department of Electrical and Computer Engineering in UT San Antonio. He has been a Xilinx, California Consultant while being as the chair at UT San Antonio. He is currently a key Global Trainer of Xilinx, California.

Artyom M. Grigoryan is an associate professor in department of electrical and computer engineering at UT San Antonio. He is an inventor and patents holder.