УДК 519.6

Parallel Fast Fourier Transform Algorithms Applications for Telecommunications

M. O. Alieksieiev
National Technical University of Ukraine "KPI", Ukraine

The article is dedicated to analysis of computationally efficient and energy saving algorithms based on Fast Fourier Transform parallel implementation in wireless telecommunications where a customer's terminals must possess such important qualities as little size and long operating time using one battery without recharging.

Keywords: *computationally efficiency, energy saving algorithms, Fast Fourier Transform, parallel implementation, wireless telecommunications, operating time.*

Стаття присвячена аналізу ефективних в обчислювальнім відношенні й енергозберігаючих алгоритмів, заснованих на швидкому перетворенні Фур'є, з точки зору їхньої реалізації для паралельного виконання в безпровідних телекомунікаціях, де термінали клієнтів повинні мати невеликий розмір і тривалий час автономної роботи від акумулятора без перезарядження.

Ключові слова: *обчислювальна ефективність, енергозберігаючі алгоритми, Фур'є швидке перетворення, паралельна реалізація, безпровідні телекомунікації, час роботи.*

Статья посвящена анализу эффективных в вычислительном отношении и энергосберегающих алгоритмов, основанных на быстром преобразовании Фурье, с точки зрения их реализации для параллельного выполнения в беспроводных телекоммуникациях, где терминалы клиентов должны иметь небольшой размер и длительное время автономной работы от аккумулятора без перезарядки.

Ключевые слова: *численная эффективность, энергосберегающие алгоритмы, Фурье быстрое преобразование, параллелизм, беспроводные телекоммуникации, время работы.*

1. Introduction

Rapid progress of telecommunication technologies has caused that user's demands and required speeds for data transmission have dramatically increased, thus leading to necessity of designing new approaches to implementing these systems into life. Being a modern trend in science, parallel computing is widely used in many applications, such as OFDM technology (used in ADSL, VDSL, IEEE 802.11a/g, DVB terrestrial digital TV systems DVB-T, DVB-H, T-DMB and ISDB-T etc, where parallel implementation of FFT is required). The market provides the telecommunications industry with dedicated FFT chips. For instance, on September 26th, 2003 SiWorks Inc., a provider of semiconductor design services and semiconductor intellectual property, announced that high throughput parallel FFT core for the emerging 802.15.3a multi-band OFDM UWB standard has been developed.

However, among all the conventional serial FFT algorithms, which used to be and continue being popular, the choice of parallel ones is very limited. In applications such as the pseudospectral methods for solving partial differential equations (PDE's), a number of multidimensional FFT's are computed per time step. The speed of the FFT computation is therefore very critical to any large application using the pseudospectral method. Since such very large computations are feasible mostly on only parallel machines, there is a need for fast multidimensional FFT algorithms for parallel machines.

2. Approaches to computing multidimensional FFT

The approach to computing multidimensional FFT's on parallel machines is currently under debate. There are two possible methods. One of the approaches is the "Transpose Method". In this method, data are divided by planes between nodes. For example, in the three dimensional transform, each node has a number of planes on which it computes two dimensional FFT's. Next, a distributed transpose rearranges the data in such a way that the FFT along the third dimension can be computed locally. This method is fairly easy and has been implemented for a number of applications.

The second approach is to design a distributed FFT algorithm which operates without collecting planes or rows on a single node. Here the internode communications are interspersed with the computation at different stages of the FFT. This algorithm is more difficult to design and implement since the parallelization is an integral part of the algorithm. It gives a clear advantage of flexibilty in data distribution, since parallelization is possible along more than one dimension [1].

For example, following approach was proposed [2] for the design of parallel FFT:

1. Designing a parallel radix 4 FFT algorithm for 16, 64, 256 and 1024 points.

2. Using a Canonic Signed Digit (CSD) representation for the multiplication coefficients.

3. Pipelining after each butterfly of the FFT algorithm.

4. Fixed point arithmetic with a variable precision, set by design parameters. Floating point arithmetic was sacrificed for reasons of speed and complexity.

And the main problems were complexity and speed of the chip, noise introduced by truncation errors in the fixed-point arithmetic also.

The described [1] algorithm can compute FFT in one, two or three dimensions for different blocksizes along different directions. The data could be parallel along one or more dimensions in any combination. A real-to complex FFT (RFFT) can be computed as a special case of this algorithm, where at least one dimension is non parallel. The non parallel dimension of the RFFT is computed within the node, and the computations for the remaining dimensions are similar to those in the CFFT algorithm.

The domain decomposition for this version is column wise; a domain consists of a three dimensional column of size $N_x \times N_y / P_y \times N_z / P_z$, where $N_x \times N_y \times N_z$ is the global data size and P_y / P_z is the processor grid. An example is shown in Figure 1 with a problem size of $8 \times 8 \times 8$ on a 2×2 processor grid.

The following outline describes an efficient parallel implementation of the FFT algorithm:

1. Chose processor grid such that Py is minimum possible.
2. Compute real to complex FFT along x.
3. Do a distributed transpose between x and y such that y becomes local and x is distributed on Py processors.
4. Compute complex to complex FFT along y.
5. Do a distributed transpose between y and z such that z is local and y is distributed on Pz processors.
6. Compute complex to complex FFT along z.
7. Do the reverse transposes and restore the data distribution.

Moreover, the distributed FFT was presented in [1]. It has two kernels: one for computations and one for communications. The computation kernel is used in each stage of the FFT calculation while the communication kernel is used only in the stages which require offnode data. Both the kernels are common to all nodes. This form of organization has the advantages of modularity and uniformity. It dissociates the housekeeping complexity of the algorithm from the two most time consuming aspects of it, thus simplifying optimization and performance analysis of the algorithm. This algorithm has been implemented on IBM SP1 housed Argonne National Laboratory.

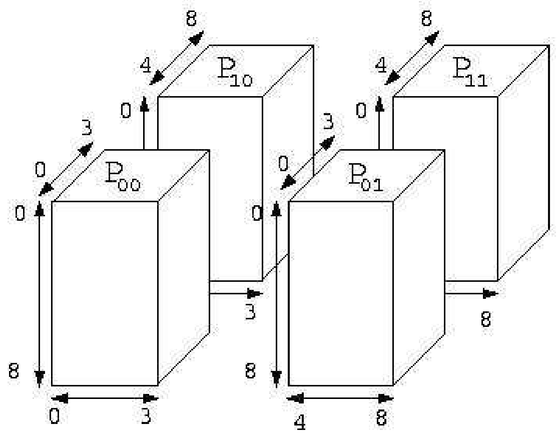

Fig. 1. Column-wise domain decomposition for 4 Processors.

4. FFT algorithms in Network-on-chip environment

Due to rapid development of a System-on-Chip (SoC) concept a special approach to designing efficient FFT algorithms for Network-on-chip Communication subsystem is required. In [5] three different parallel FFT algorithms are presented. There are three main steps to be performed running the algorithms:

- preprocessing (data is rearranged in bit-reverse order and divided by p blocks)
- actual transformation (transform is executed further decomposition into sequential execution and parallel execution can be performed
- post processing.

The proposed algorithms maximize the data parallelism as well as minimize the communication overhead resulting in higher performance in multiprocessor SoC.
Not going deep into the principles of the aforementioned algorithms, I would like to confine myself to presenting the simulation results from [5]. The speed comparison of the three methods is given below.

As the number of points in FFT, N increases, the speedup ratio reaches the number of processing elements. As it can be seen on the graphs methods 2 and 3 show higher speed in comparison with the first method.

The proposed algorithms show the improvement over the existing parallel algorithms for the following reasons:

- the resources are used in a balanced manner so that the divided task for each processing element finishes in shorter time
- the data locality is also well utilized to minimize the communication load
- the concurrency of communication and computation in Network-on-Chip environment can hide communication overhead.

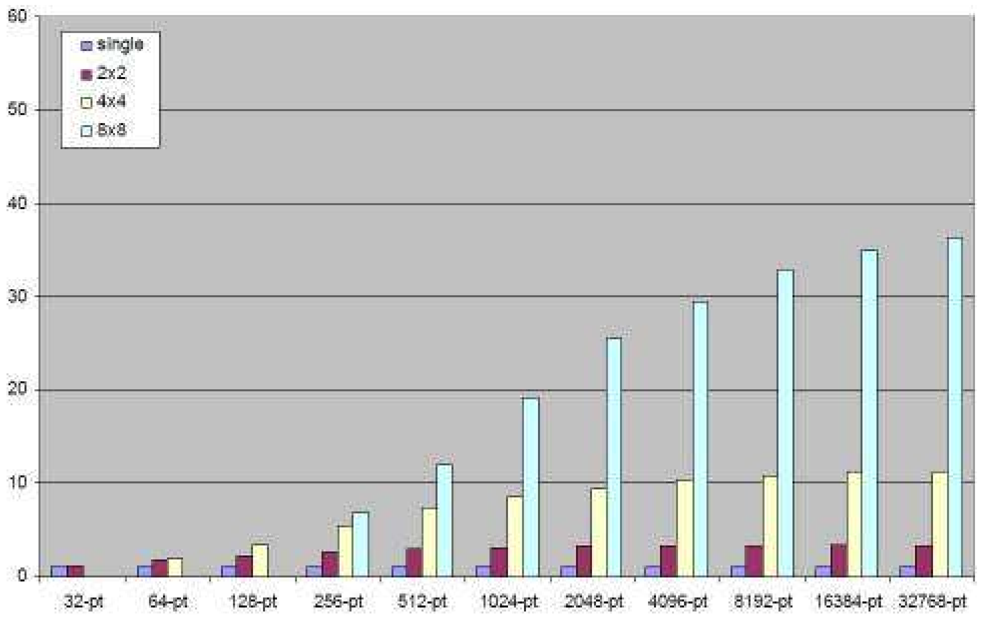

Fig. 2. Method one. [5].

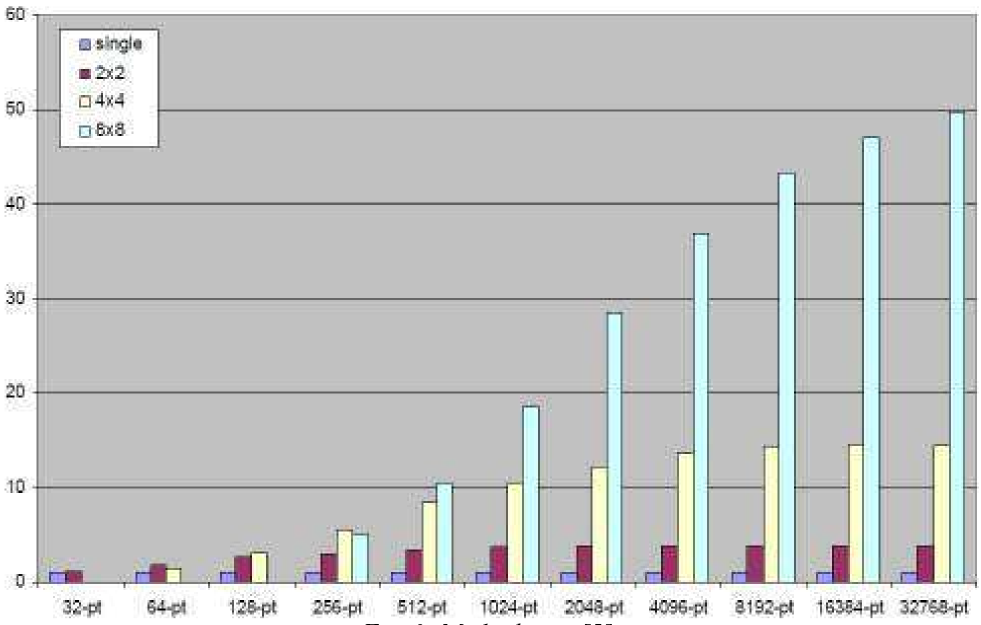

Fig. 3. Method two. [5].

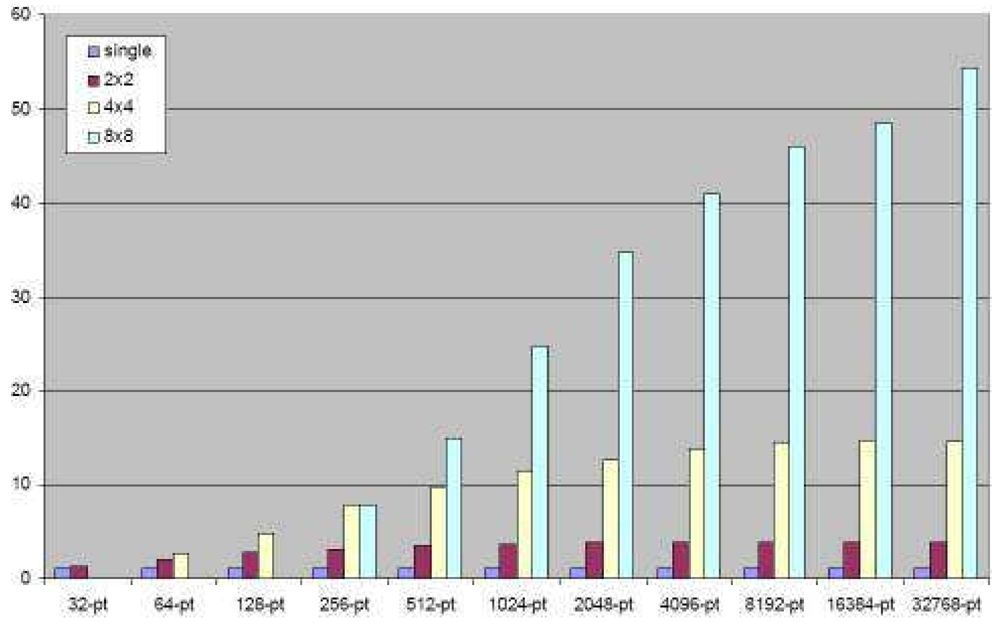

Fig. 4 Method 3. [5]

5. Parallel FFT Algorithm on Multiprocessors with Cache Technology

As far as multiprocessor system is widely used, it is reasonable to consider a high-performance parallel FFT algorithm which efficiently works in a multiprocessor environment. In [4, P. 105] an interesting concept of such an algorithm is presented. The main idea is to use the intrinsic property of FFT parallelism for a parallel algorithm. In addition, the role of cache is also considered.

In figure 4 the FFT computation parallelism property can be observed.

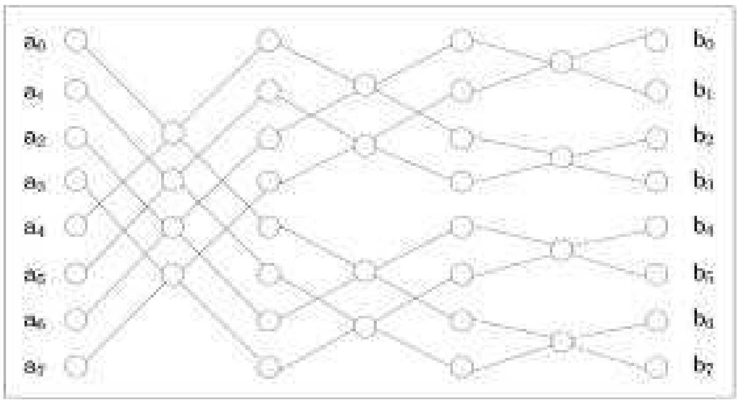

Fig. 5. FFT computation parallelism property [4].

Below in the figure 6 the simulation results from [4, P. 105] are given. The proposed FFT method is compared with 2 conventional DSP methods (TS101 and C6701).

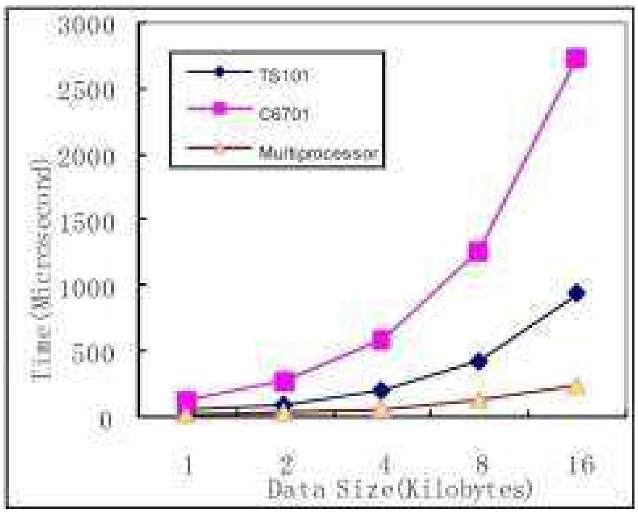

Fig. 6. FFT Methods comparison. [4]

It can be observed that the proposed algorithms outperforms its rivals in terms of speed and productivity.

6. Mapping Parallel FFT Algorithm onto SmartCell Coarse-Grained Reconfigurable Architecture

Authors in [8, P. 231-234] argue that instead of using the aforementioned algorithms mapping parallel FFT Algorithm onto SmartCell device can be used. It is claimed that this approach allows to outperform NoC concept (namely, it is 2.7 times faster).

Energy consumption is also the problem to be considered. In figure 7 it is shown how the suggested novel approach outperforms the FPGA concept.

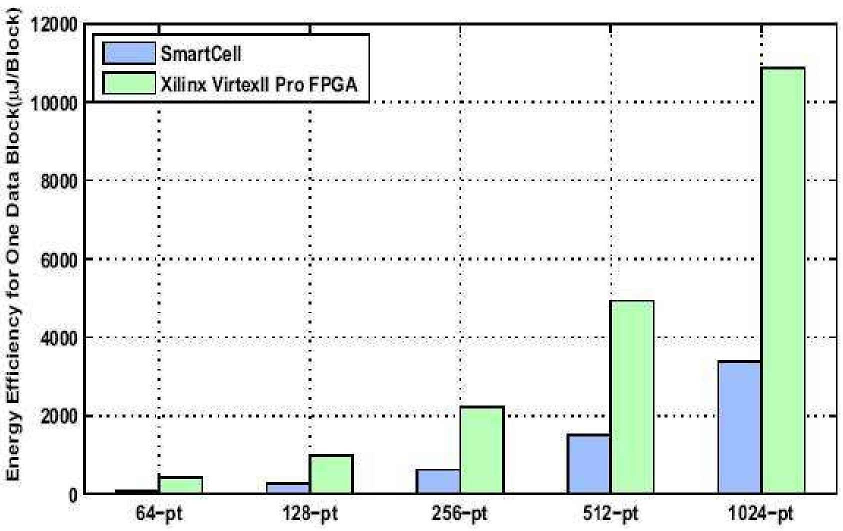

Fig. 7. Energy consumption comparison between SmartCell and FPGA. Taken from [5].

7. Partial FFT

Instead of processing all the input data partial use of the given data can be considered. In [7, P. 1604-1615], for instance, pruning the useless data flow is suggested, so that a significant speed up can be achieved. This concept is especially relevant for streaming applications.

A simple example of data flow graph in Partial FFT is depicted in fig. 8.

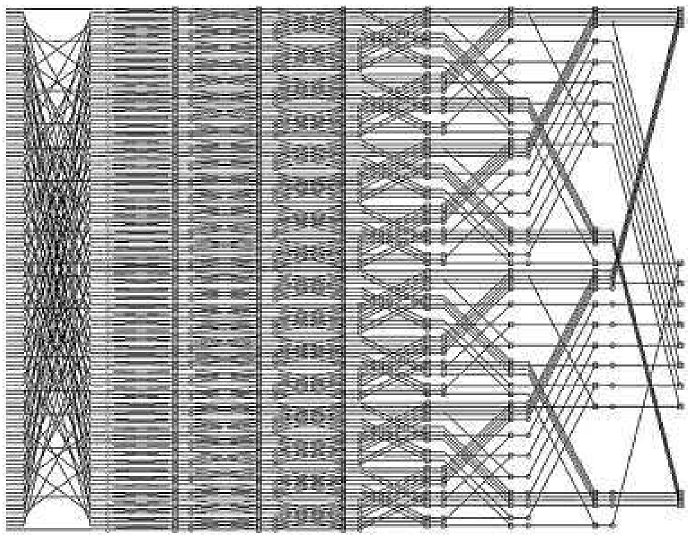

Fig. 8. A simple example of data flow graph

Simple example of data flow graph in PFFT. 17 out of 128 output bins are used. The corresponding FFT is a standard 128-point radix-2 Cooley–Tukey decimation-in-time variant. Taken from [7, P. 1604].

8. Conclusions

Provided analysis shows that there are a lot of technologies in telecommunications that use FFT as a key point. Which particular algorithm to use depends on the system developer. However, it is clear that usage of multi-core processors is extremely efficient due to high performance of the parallel computations and due to its energy-saving characteristic. However the results of the analysis are such that there is no unanimous approach to parallel FFT implementation. Thus its development is very challenging especially in terms of further implementation in future telecommunications systems.

REFERENCES

1. Dubey, A.; Clune, T. Optimization of a parallel pseudospectral MHD code. Frontiers of Massively Parallel Computation, 1999. Frontiers apos;99. The Seventh Symposium on the Volume , Issue , 21-25 Feb 1999 Page(s):208 – 212.
2. Roland Weigand. Design of a parallel FFT processor using fixed point arithmetic and CSD-Multiplication. - ESA/ESTEC, XRM Section, 1995, *Pages: 132 – 136.*

3. Strong, David M.; Magee, Eric P.; Lamont, Gary B. Implementation and test of wave optics code using parallel FFT algorithms. - Air Force Institute of Technology, 2001, *Pages: 344-351.*
4. Jun Tan, Xingshu Chen, Long Xiao. An Optimized Parallel FFT Algorithm on Multiprocessors with Cache Technology in Linux.. 2008 International Symposium on Computer Science and Computational Technology, Pages: 105-109.
5. Jun Ho Bahn, Jungsook Yang and Nader Bagherzadeh Parallel FFT Algorithms on Network-on-Chips. Fifth International Conference on Information Technology: New Generations, Pages: 1087-1093.
6. P. Dmitruk, L.-P. Wang, W.H. Matthaeus, R.Zhang, D. Seckel. Scalable Parallel FFT for spectral simulations on a Beowulf cluster. Parallel Computing, #27, Pages: 1921-1936.
7. Min Li, David Novo, Bruno Bougard, Member, IEEE, Trevor Carlson, Liesbet Van Der Perre, Member, IEEE, and Francky Catthoor, Fellow, IEEE. Generic Multiphase Software Pipelined Partial FFT on Instruction Level Parallel Architectures. - IEEE Transactions On Signal Processing, Vol. 57, No. 4, April 2009, Pages: 1604-1615.
8. Cao Liang and Xinming Huang. Mapping Parallel FFT Algorithm onto SmartCell Coarse-Grained Reconfigurable Architecture. 2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors, Pages: 231-234.

Надійшла у першій редакції 01.04.2009, в останній – 31.10.2009.