iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** IV **Month of publication:** April 2018

DOI: http://doi.org/10.22214/ijraset.2018.4629

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue IV, April 2018- Available at www.ijraset.com

An Optimization approach for Artificial Neural Network based Co-Channel-Interference Reduction Scheme using Commodity Video Cards

Umiya Mushtaq[1], Col. Dr. O P. Malik[2], Mohammad Atif [3], Barkat Hafeez[4]
[1, 2, 3, 4] *Department of* Electronics & *Communication Engineering, Al-Falah University, India*

Abstract: ***Many communication system problems are solved using Artificial Neural Network. However, many of these modern applications demand faster processing for high performance applications, where traditional ANN cannot be used because they are very slow. One such example is CCI or co-channel interference, which is a result of frequency-reuse in a cellular network. Many expensive alternatives have been suggested for speeding up the processing. This includes using ASICs and FPGAs. In this work we describe our research outputs regarding optimization of Artificial Neural Network based Co-Channel-Interference using commodity Video Cards. Our Optimization approach is based on a parallel algorithm for Firefly Algorithm's fitness function, and is tailor made for a GPU or a graphical processing unit, which is the main part of a video card. We tested our implementation on a graphical processor having 192 cores. Our proposed fast optimized system provides a very economical solution as compared to ASICs or FPGAs. The resulting implementation is simple and scales up with the processor cores.***
Keywords: Video Cards, NVIDIA CUDA, ANN, Wireless, CCI, Firefly Algorithm, Fitness Function

I. INTRODUCTION

In mobile radio environment, there are many challenges. Co-channel interference (CCI) resulting from frequency reuse is one of those issues. There are many other issues such as due to the dynamic nature of the channel, fading is a common problem ANN has been applied in smart antenna systems to reduce CCI. ANN based smart antenna systems is expected to perform well for computationally expensive signal processing for CCI reduction. Neural networks, once trained and validated, are not only capable of performing computationally complex signal processing but also can track sources in real time and are nonlinear in nature. These characteristics make them suitable for many communication system related problems including CCI reduction. Moreover if we use a parallel hardware like a video card for implementing an ANN we can achieve maximum speed up [8]. A video card has a Graphical Processing Unit or a GPU which is used to accelerate graphics related jobs.

Many programming platforms have been developed for GPU programming. NVIDIA's Compute Unified Device Architecture (CUDA) [7] is one of the most common ways to program the GPU. The traditional approach before CUDA has a very slow learning rate. CUDA can be extremely helpful in accelerating ANN algorithms on GPUs [8] [1]. CUDA is C language with some extensions for processing on GPUs. The user writes a C code, while the compiler bifurcate the code into two portions. One portion is delivered to CPU (because CPU is best for such tasks) while the other portion, involving extensive calculations, is delivered to the GPU(s) that executes the code in parallel. With such a parallel architecture, GPUs [5] provide excellent computational platform, not only for graphical applications but any application where we have significant parallelism. Since an ANN has a number of processing nodes or Neurons. These Neurons work independently. Thus each of these processors can work independently. Therefore, we intend to do this processing in parallel on Graphical Processing Unit having 100s of processor cores. In this work we utilized NVIDIA CUDA parallel programming platform on CUDA- Compatible GPU. To develop an Artificial Neural Network on a Graphical Processing Unit ,we used NVIDIA CUDA for its implementation. GPUs have hundreds of processing units and have a highly parallel architecture , that clearly maps to ANN as ANN is also a massively parallel system. In addition to this the GPU-based ANN is much more cost effective as compared to FPGA or ASIC based solutions. The main idea is to do the CCI calculations faster by availing the parallelism of ANN and implementing it on paralle hardware like GPU and thus accelerating it for real-time communication. This research aims at implementation ANN for smart antenna design on a GPU in order to improve the performance as compared to CPU implementation.

Thus CUDA accelerated ANN algorithms can be used in many real-time applications, including CCI reduction in mobile communications ,Image processing, character recognition ,object classifications, voice recognition and in a number of systems which require intelligence and auto control.

Thus CUDA accelerated ANN algorithms can be used in many real-time applications, including optimization of CCI reduction in mobile communication, Image processing, object classifications, voice recognition and in a number of systems which require intelligence and auto control. The GPU's resources that are ALUs can be used by many applications. The GPU also have ECC (Error correction) capabilities which are very important for scientific applications. In graphics many a times error correction capabilities are not much required, but in case of scientific applications like our own ANN application accuracy is very important and error correction capability must be there. All models of a GPU are now coming up with ECC capabilities. GPU computing is the use of a GPU (graphics processing unit) to do general purpose scientific and engineering computing. The model for GPU computing is to use a CPU and GPU together in a heterogeneous computing model. The sequential part of the application runs on the CPU and the computationally-intensive part runs on the GPU [17]. From the user's perspective, the application just runs faster because it is using the high-performance of the GPU to boost performance. In the rest of this paper we discuss our idea for speeding-up the CCI reduction process, and also discuss implementation details. First we briefly describe how an ANN problem can be mapped to GPU architecture, in general. After that we describe our approach for computation of Firefly Algorithm, fitness function.

II. EVOLUTION OF MULITCORE ARCHITECTURE

An ANN has several layers. And in a particular layer ANN has a number of processing nodes or Neurons. These Neurons work independently, which effectively means each of these processors can work in parallel. However, if we perform these processing on a CPU i.e. the large amount of calculations of ANN, it will consume lot of time, making it unsuitable for real-time processing. Because of this we planned to do ANN processing in parallel on a Video card, i.e. on a GPU. A GPU has 100s of processor cores as compared to a CPU with very few cores.

In this work we utilized NVIDIA CUDA parallel programming platform on CUDA-Compatible GPU. The CPU are now coming up with multicore since the year 2003.Till 2003 we had single Core in the CPU but after the year 2003 the trend started for putting more and more cores even in a CPU. The basic reason of putting these cores is that we were not able to increase the speed of the CPU by simply increasing the clock frequency norm that was in place for the last 40 years. This is because if we increase the clock frequency the power dissipation will increase. So CPU designers put more cores instead of increasing the clock frequency. Actually when clock frequency is higher the transistors in the CPU will switch on and off faster resulting in much more dynamic power dissipation. This would obviously give poor power backup and also will generate lot of heat on the surface of the chip so we will have to improve the power dissipation mechanism. That will make the system more costly and obviously will give less power backup which we don't want. Most of the CPUs which we are using today are being utilized to run Complex applications even in embedded systems. In embedded systems we have lot of constraints on the power budget and also on the computation and on the memory. An embedded system is basically a very constraint system. On the other hand a GPU has been having lot of cores since its invention.

Basically these many cores in a GPU were put because it has to do lot of calculation for graphical processing [4,6,10]. But the researchers realized that these cores can also be used for performing non graphical applications. This field is called GPGPU. The GPGPU means general purpose programming on graphical Processing Unit.

Eventually problem of artificial neural network is obviously a non graphical application. Our implemented artificial neural network is running on a GPU in its hundreds of cores and these are running in parallel. Because these are computed in parallel these are expected to be faster as compared to the serial applications[18,4,6,10].

Now a days performance is of prime importance in many applications, because we want better applications e.g. better GUI, the graphical user interface, and maybe faster Antivirus and in faster drug Discovery. In all these examples performance plays a key role. If these applications are slow these will not meet the uses of the customer expectations. The performance Goal can be met provided we do parallel computation on a CPU and now on a GPU thanks to NVIDIA CUDA.

We can see that now because all the computers are becoming parallel more parallel applications are coming up in the market and we expect the growth of the parallel softwares more and more. This also means the demand of parallel programmer should keep on increasing.

III. THE MAIN IDEA BEHIND USING AN ANN FOR CCI AND OUR APPROACH

In [1] the authors have described a few approach for ANN implementation for channel equalization. One of the approach is using Firefly Algorithm or FFA. The main idea behind using an ANN for CCI reduction in particular and in other problems in general is the minimization of a Fitness Function, which guarantees an optimal solution.

The fitness function is as given below:

$f(t_1) = 1/1+MSE$

$$= \frac{1}{1+1/Q\sum_{K=i}^{Q}\|d(k)-y(k)\|\hat{}2} \qquad \text{---------(1)}$$

IV. WHAT IS A GPU?

In the below paragraphs we explain how a GPU is used actually for ANN implemnetaion and how we optimized the above fitness function..

Any particular layer in ANN has a number of processing nodes or Neurons. These Neurons work independently [11]. That means each of these processors can work independently. Therefore, we intend to do this processing in parallel on Graphical Processing Unit having 100s of processor cores. In this work we utilized NVIDIA CUDA parallel programming platform on CUDA-Compatible GPU. Most software developers have relied on the advances in hardware to increase the speed of their applications under the hood; the same software simply runs faster as each new generation of processors is introduced. This drive, however, has slowed since 2003 due to energy consumption and heat-dissipation issues that have limited the increase of the clock frequency and the level of productive activities that can be performed in each clock period within a single CPU. Virtually all microprocessor vendors have switched to models where multiple processing units, referred to as processor cores, are used in each chip to increase the processing power. This switch has exerted a tremendous impact on the software developer community [Sutter 2005].

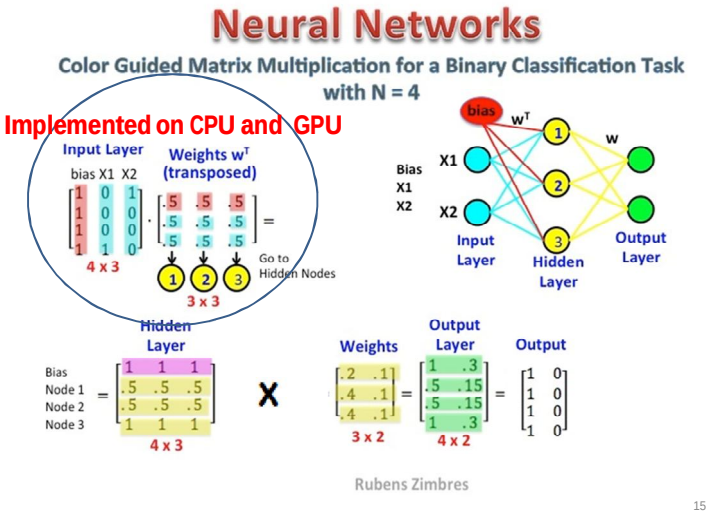


Figure 1: A general example of Neural Network operations

Traditionally, the vast majority of software applications are written as sequential programs, as described by von Neumann [1945] in his seminal report. The execution of these programs can be understood by a human sequentially stepping through the code. Historically, computer users have become accustomed to the expectation that these programs run faster with each new generation of microprocessors. Such expectation is no longer strictly valid from this day onward. A sequential program will only run on one of the processor cores, which will not become significantly faster than those in use today.

Without performance improvement, application developers will no longer be able to introduce new features and capabilities into their software as new microprocessors are introduced, thus reducing the growth opportunities of the entire computer industry. Rather, the applications software that will continue to enjoy performance improvement with each new generation of microprocessors will be parallel programs, in which multiple threads of execution cooperate to complete the work faster. The high-performance computing community has been developing parallel programs for decades. These programs run on large-scale, expensive computers.

Now that all new microprocessors are parallel computers, the number of applications that must be developed as parallel programs has increased dramatically. There is now a great need for software developers to learn about parallel programming.

Artificial Neural Network can be represented as a cluster of parallel processors as shown below in figure.

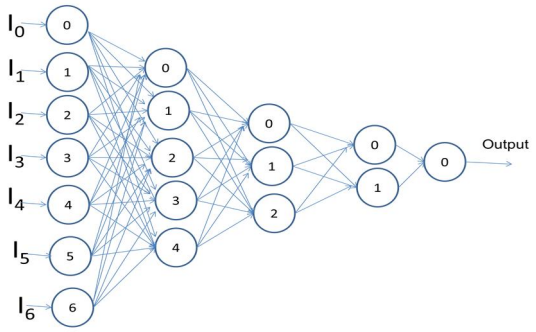


Figure 2: A Neural Network consist of many layers of Neurons

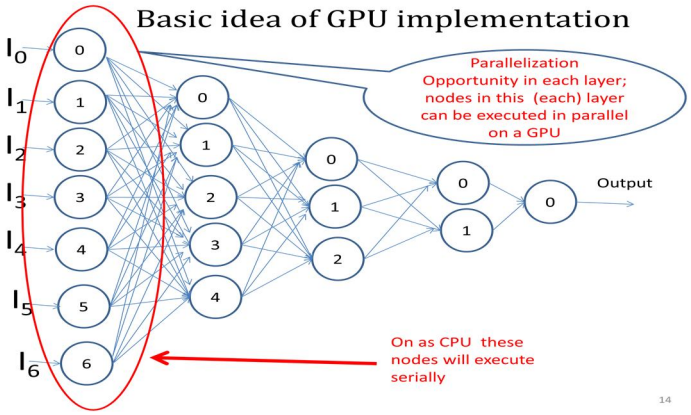


Figure 3: Basic idea behind implementation of Neural Network on a GPU

We implemented these complex structures and we found very good performance on GPU.

A. Assumptions

1) We assumed that the network is already trained and thus we have a file where the weight sets are stored.
2) We exploited the parallelism in calculating the output at each Neuron in each layer.
3) The performance was compared at each layer output and also at the final output.

From the above discussions we conclude that an ANN can be modeled by matrix equation as shown in following figure:

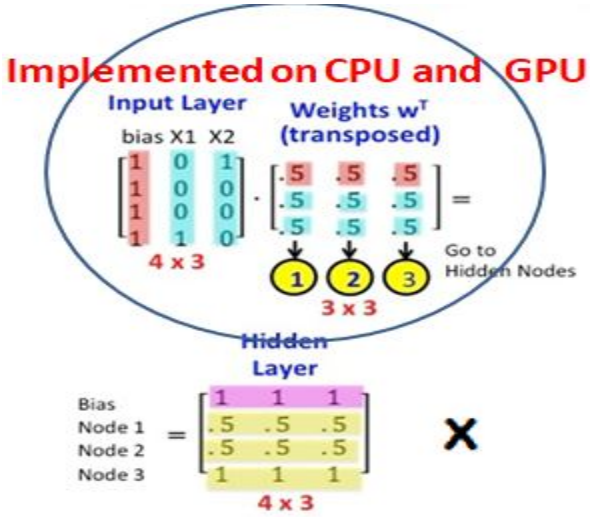


Figure 4: Neural Network operations at the output is basically a matrix multiplication operation

Thus we can obtain the output of ANN by matrix multiplication.
We now give more details on our optimization approach. We observe the denominator in the fitness function, and we term it as

B. Neuro Reduction

$$f(t_1) = 1/1+\boldsymbol{MSE} = \frac{1}{1+1/Q\sum_{K=i}^{Q}\|d(k)-y(k)\|^2}$$

NeuroReduction()

The concept has been further explained in below figures:

$$f(t_i)=\frac{1}{1+MSE}=\frac{1}{1+\frac{1}{Q}\sum_{k=1}^{Q}\|d(k)-y(k)\|^2}$$

NueroReduction();

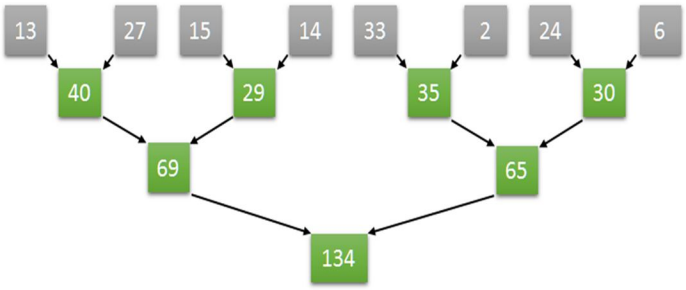


Figure 5: NueroReduction

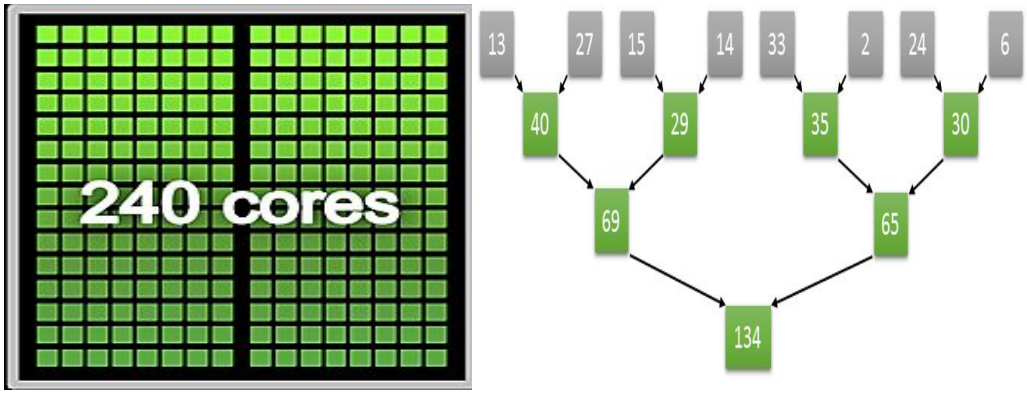


Figure 6: NeuroReduction is performed on a GPU in parallel

Following figure shows the two most computationally expensive jobs we offloaded to GPU:

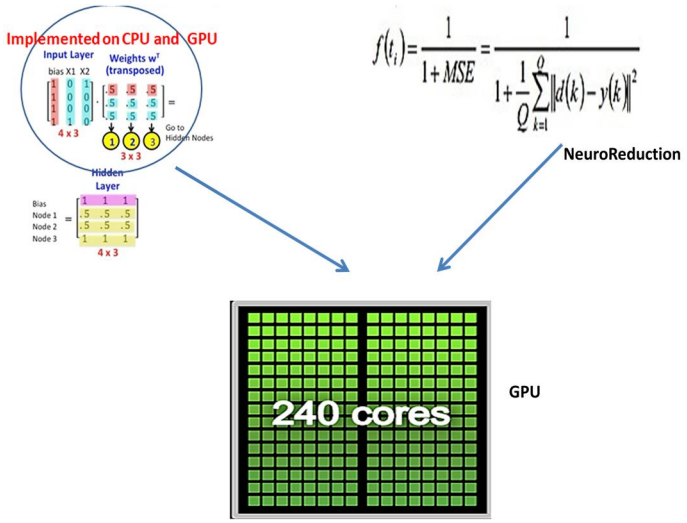


Figure 7: The two operations best suited for GPU implementation of ANN based CCI reduction scheme

The above figure summarizes our approach. We have used GPU for offloading two computationally expensive jobs from ANN based for CCI reduction scheme. As explained in previously those two jobs are:

1) Output calculation at each layer

This operation basically comes out as a matrix multiplication operation. Matrix multiplication is indeed a good problem to be solved on a GPU as it bears high Arithmetic Intensity.

2) Fitness Function calculation in FFA : A closed observation makes it clear that one of the most computationally expensive jobs here is the summation term in the denominator of the Fitness Function. We can express this term as a SUM reduction operation. The Sum reduction has been explained in the Figure 4. CPU this operation will be performed serially using a single thread. But on a GPU we performed this operation using a 100s of processing threads in parallel.

V. RESULTS & DISCUSSIONS

The results of the two steps are given in the following tables. The reported performance gain in terms of speed up has been tested on a GPU with 240 cores. The same operations have been run on a CPU only system. The speed up as been shown in the last column

Table-I Output Calculation At Each Layer

S, No.	No. of Neurons in the layer	Speed-up
1	10	0.1
2	100	2.0
3	200	2.5
4	300	2.9
5	400	3.6

Table Ii Fitness Function Calculation In Ffa

S, No.	No. of Training data points	Speed-up
1	100	0.06
2	512	3
3	1024	8
4	2048	10
5	4096	50

From both the tables it is clear that when the problem size is small, there is actually performance degradation. It is only when we have lot of data that the GPU results in good performance.

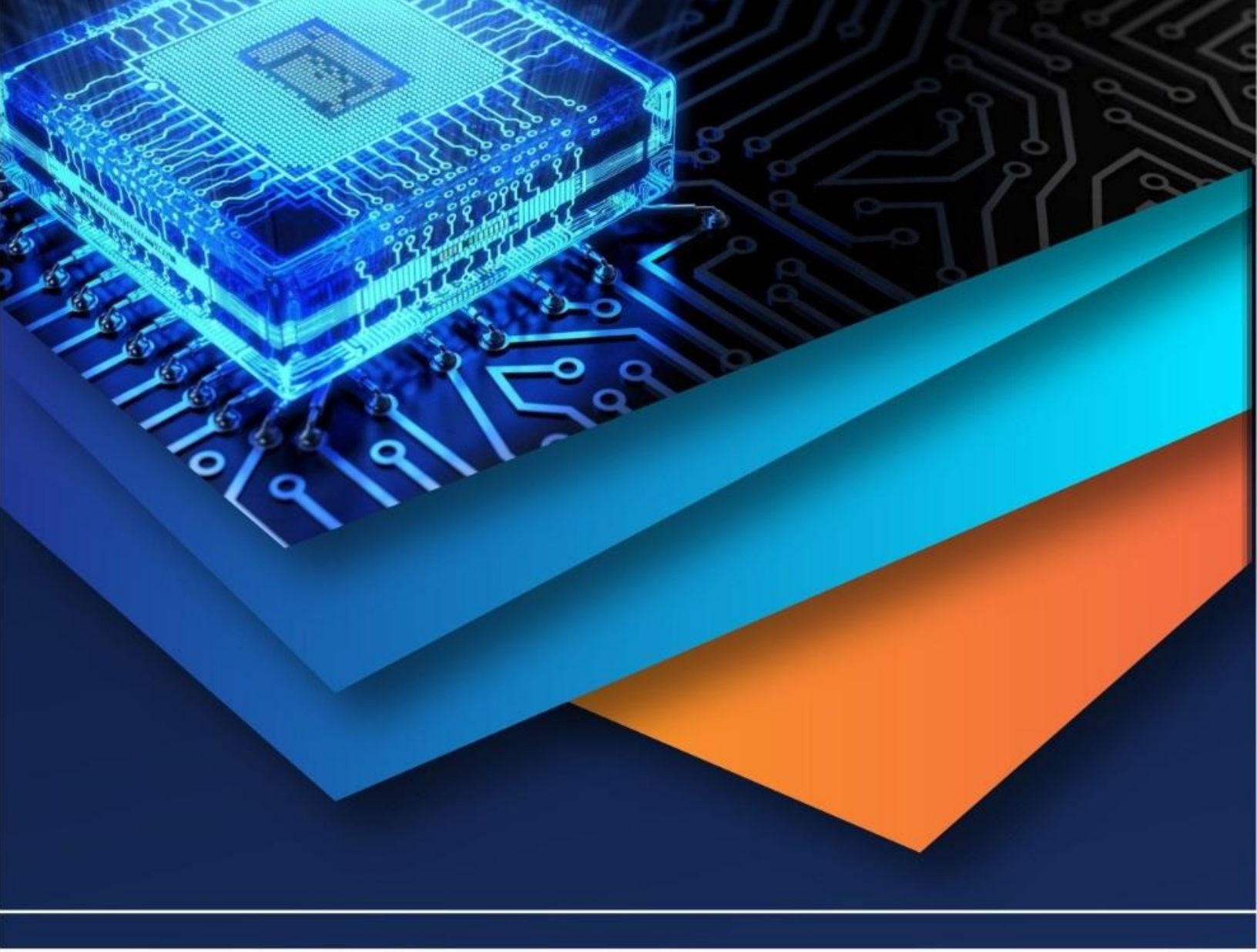
VI. CONCLUSION

In this paper we have explored an approach for optimizing the ANN based Co-Channel Interference reduction scheme reported in [1]. This approach uses FFA and its modified version. We identified two areas where the reported algorithm can be further improved and can be speed up. Those two are termed as Output calculations and Neuro Reduction function. For CCI reduction these two stages are performed and cannot be avoided. However these are two most expensive and consume lots of CPU time thereby making it unsuitable for any meaningful real-time application. Our Initial kernels have reported a performance gain of around 50 times in case of Neuro Reduction.

In this paper have also explored exactly what are the areas in Artificial Neural Network that can be implemented on a GPU. While other approaches have been presented in literature [14][15], our approach is different since we are using an economical platform for accelerating ANN computations in real-time or near real-time. We have also explained what are the advantages of using a GPU, namely GPU is a parallel platform where all the Neurons in a particular layer can be implemented and the neurons would do all the calculations in parallel on so many cores of a GPU. There are many advantages of GPU based implementation one of which is that it is an economical solution as compared to other proposed solutions such as using FPGA and ASICS. Further the development time of GPU based implementation would be much lesser as compared to an FPGA and ASIC based system so the time to market would be much less as compared to other solutions. Of course one of the challenges in GPU programming is the availability of parallel algorithms. In short we have to find the areas in our algorithm where we can apply the GPU and we have to modify the algorithm so that it suits best for the GPU.

REFERENCES

[1]. Padma et al, Neural Network training using FFA and its variants for Channel Equalization, International Journal of Computer Information Systems and Industrial Management Applications. ISSN 2150-7988 Volume 9 (2017) pp. 257-264
[2]. David B. Kirk and Wen-mei W. Hwu, Programming Massively Parallel Processors: A Hands-on Approach by Morgan Kaufmann (February 5, 2010),ISBN 0123814723
[3]. R. H. Luke, D. T. Anderson, J. M. Keller S. Coupland, “Fuzzy Logic-Based Image Processing Using Graphics Processor Units”, IFSA-EUSFLAT 2009
[4]. Pat Hanrahan , “Why are Graphics Systems so Fast? ” , PACT Keynote, Pervasive Parallelism Laboratory, Stanford University, September 14, 2009
[5]. Shuai Che , Michael Boyer, Jiayuan Meng, David Tarjan , Jeremy W. Sheaffer, Kevin Skadron, “A Performance Study of General-Purpose Applications on Graphics Processors Using CUDA”, Journal of Parallel and Distributed Computing, ACM Portal, Volume 68 ,Issue 10, pp. 1370-1380 October 2008
[6]. Kayvon Fathalian and Mike Houston, “A closer look at GPUs”, Communications of the ACM, Vol . 51 no. 10, October 2008
[7]. R. Tom, Halfhill, “Parallel processing with CUDA”, Microprocessor report. January 2008
[8]. Z. Luo, H. Liu and X. Wu, “Artificial Neural Network Computation on Graphic Process Unit”, Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference, vol.1, pp. 622 – 626, 31 July-4 Aug. 2005
[9]. Araokar, Shashank, “Visual Character Recognition using Artificial Neural Networks” Cog Prints, May 2005
[10]. Trendy R. Hagen, Jon M. Hjelmervik, Tor Dokken, “The GPU as a high performance computational resource”, Proceedings of the 21st spring conference on Computer graphics, pp. 21 – 26, 2005
[11]. K. Oh, “GPU implementation of neural networks” Pattern Recognition, pp. 1311-1314. Vol. 37, No. 6. (June 2004)
[12]. Earl Gose Steve Jost, Richard Johosonbaugh, ”Pattern Recognition and Image Analysis” , Eastern Economy Edition, Prentice Hall ,1999.
[13]. Simon Haykin, “Neural Networks: A comprehensive foundation”, 2nd Edition, Prentice Hall, 1998
[14]. Neural Networks in Smart Antenna Design for Co-channel Interference (CCI) Reduction: A Review
[15]. Parallel Training of Multilayer Perceptron on GPU. -Chris Oei ,,Gerald Friedland ,Adam Janin]
[16]. Victor Podlozhnyuk, Mark Harris , “NVIDIA CUDA Implementation of Monte Carlo Option Pricing”, Tutorial in NVIDIA CUDA SDK 2.3,
[17]. “CUDA Programming Guide 3.0”, Published by NVIDIA Corporations.
[18]. Elizabeth, Thomas , “GPU GEMS: Chapter 35 Fast Virus Signature Matching on the GPU”, Addison Wesley



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY