iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 6 **Issue:** V **Month of publication:** May 2018

DOI: http://doi.org/10.22214/ijraset.2018.5199

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 V, May 2018- Available at www.ijraset.com

Energy Efficient Computation Method for CPU-GPU System on Chip

C. Sai Punitha[Q1], D. Jessintha[2]
[1, 2]*Easwari Engineering College*

Abstract: ***The increasing trends in multi-core chips allows higher performance at lower energy and the communication between the cores is a limiting factor which can be improved by the parallel computation such as thread level parallelism. The improvement in performance gained by the use of a multi-core processor depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can run in parallel simultaneously on multiple cores. One such algorithm called Double Modulus Number Theoretic Transform finds good for large amount of data computational methods especially with zero round off errors which provides high efficiency in communication between heterogeneous multicores The parallelization algorithm Double Modulus Number Theoretic Transform is computed and developed with software development platform CUDA especially design for its Nvidia Graphic cards which provides higher bandwidth and speed with less computational complexity.***

I. INTRODUCTION

Number Theoretic Transforms (NTT), made their appearance and with an aim of replacing the FFT as a tool of signal processing. These transformations based on rules of the number theory. However, in spite of the advantages of this transformation (exactness of the results), the NTT did not know a success similar to that of the FFT because of the interpretation problem of the field that it create, which is not the case for the frequency field created by the FFT. This transformation bases on the modulo-M properties. Hence, the good choice of M may gives properties similar to those of the Discrete Fourier Transform. The NTT is a generalization of the classic DFT to finite fields. With a lot of work, it basically lets one perform fast convolutions on integer sequences without any round-off errors, guaranteed. Convolutions are useful for multiplying large numbers or long polynomials and the NTT is asymptotically faster than other methods like Karatsuba multiplication and it shows that Number Theoretic transform effective for large data computation and for high bandwidth applications. Hence, it can be incorporated in multi cores such as graphic cards in which their bandwidth limited operation and fast parallel operation has been a great impact factor multi cores processing unit. Double Modulus NTT provides the best implementation in CPU-GPU operations
with their high level parallelism and enhances the faster communication between the multi core with software development CUDA architecture in Nvidia Graphic cards which supports languages like C,C++,Python and Fortran.

II. RELATED WORKS

The paper[1] shows the two implementations of the fast Fourier transform decomposed into vector operation appropriate for cases where data to be transformed is stored in an unorthodox order and performed a vector FFT were implemented for TigerSHARC DSP and NVIDIA CUDA platforms and [3] shows the implementation of the FFT transform in graphic cards by using the Open Computing Language (OpenCL) the GPU is more promising for large number of FFT's of large sizes The results also confirm that the FPGA based implementation is faster than the built-in IP-core modules of Xilinx and Sparse Fourier Transform demonstrated in [4] achieves speedups of up to three times a single-threaded SFFT while a GPU-based version achieves up to ten times speed up the process.
In [5] a new Mersenne number transform (NMNT) can be easily parameterized and implementation in an XC2V4000 FPGA chip has shown that this architecture can work at a frequency of up to 114MHz with a throughput rate of 228MS/s. Based on the qualitative and quantitative analyses are provided to show the effectiveness of proposed NTT architectures[6] sates that the NTT architecture provides a good performance for long integer multiplication and [7] provides a Fractional Number Theoretic Transform can be efficiently applied to secure signal image processing and the testing speed of FFT/IFFT on Geforce 680m their precision errors been compared [8]and [9] incorporation of double modulus technique in NTT achieves a million bit integer multiplication for cryptographic applications and [10] incorporation of NTT on kepler architecture of graphic cards reveals the NTT can be significantly used in larger bit integer multiplication which is the first implementation of Number Theoretic Transform on graphic cards so far and the thread level parallelism has been has been implemented.

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue V, May 2018- Available at www.ijraset.com

III. OVERVIEW OF GPU

CUDA is NVIDIA's parallel computing architecture that enables dramatic increases in computing performance by harnessing the power of the GPU (graphics processing unit).With millions of CUDA-enabled GPUs sold to date, software developers, scientists and researchers are finding broad-ranging uses for CUDA, including image and video processing, computational biology and chemistry, fluid dynamics simulation, CT image reconstruction, seismic analysis, ray tracing, and much more. Computing is evolving from "central processing" on the CPU to "co-processing" on the CPU and GPU. To enable this new computing paradigm, NVIDIA invented the CUDA parallel computing architecture that is now shipping in GeForce, ION Quadro, and Tesla GPUs, representing a significant installed base for application developers. CUDA has been enthusiastically received in the area of scientific research. With the recent launches of Microsoft Windows 7 and Apple Snow Leopard, GPU computing is going mainstream. In these new operating systems, the GPU will not only be the graphics processor, but also a general purpose parallel processor accessible to any application.

A. CUDA Programming model

CUDA parallel computing architecture includes a CUDA C compiler support for OpenCL and Direct Compute which architecture to natively support multiple computational interfaces(Standard languages and APIs) as in figure 1.

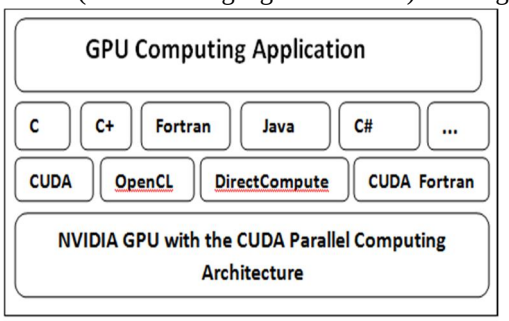


Figure 1.CUDA Architecture

The operation between CPU and GPU can be done by copying the input data from CPU memory to GPU memory, Loading GPU program and executing caching data on chip for performance and copying results from GPU memory to CPU memory.GPU executes a kernel of many threads.

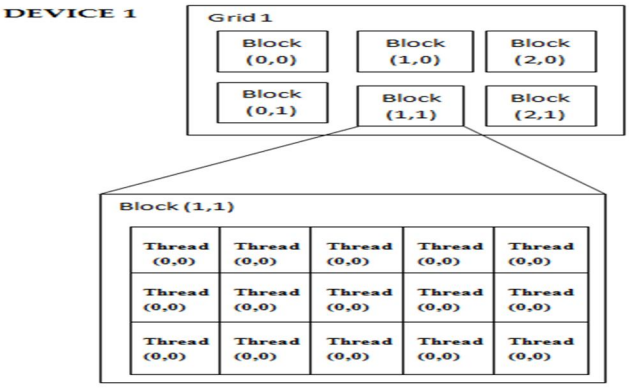


Figure 2.Architecture for CUDA programming model

As shown in figure 2 Threads are grouped into blocks and are grouped into a grid. A kernel is executed as a grid of blocks of threads. Parallel code (kernel) is launched and executed on a device by many threads and they are grouped into thread blocks parallel code is written for each thread is free to execute a unique code path of Built-in thread and block ID variables. Threads launched for a parallel section are partitioned into thread blocks Grid is equal to all blocks for a given launch thread block is a group of threads that can synchronize their execution Communicate via shared memory. Threads 3D IDs, unique within a block 2D IDs, unique within a grid Dimensions set at launch time Can be unique for each section Built-in variables such as threadIdx, blockIdx blockDim, gridDim Any possible interleaving of blocks should be valid presumed to run to completion without pre-emption can run

©IJRASET: All Rights are Reserved

in any order can run concurrently OR sequentially Blocks may coordinate but not synchronize shared queue pointer, OK shared Thread blocks can run in any order Concurrently or sequentially Facilitates scaling of the same code across many devices.

B. CUDA Memory model

CUDA processors have multiple types of memory available to the programmer and to each thread are Global memory has large address space, high latency (100x slower than cache) Shared memory has small, low latency. Texture/Constant memory has read only operations. Registers/Local memory only available to one thread Register memory, Scalar variables (e.g., int i, float f, etc.) are stored in fast registers. There are a limited number of registers per thread (although each SIMD processor has 32,768 32-bit registers) Kernel-declared arrays can also be in registers, but only if the indexing is known at compile-time. Registers are private to the thread has no sharing Read/Write, no synchronization necessary. Registers can spill into "local memory," which is just (slow) global memory set aside for each thread. New GPUs do have caches for local memory. if the registers have been spilled is to look at the PTX code for the ".local" mnemonic. Read/Write, no synchronization necessary.

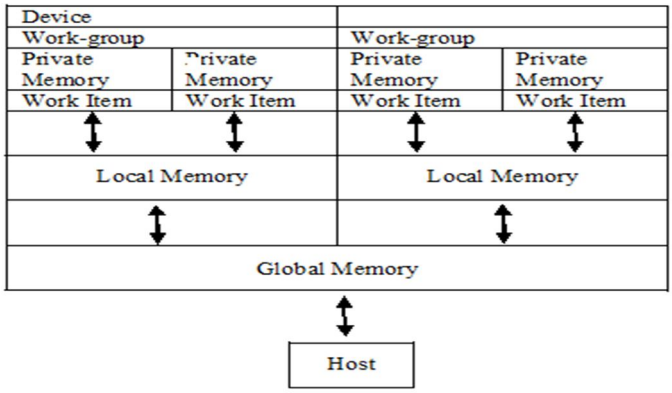


Figure 3.Architecture for CUDA memory model

Shared Memory Variables declared with __shared__ are stored in shared memory, which is very fast. It is, however, limited (48KB per multiprocessor on our GPU) has the lifetime of a block can be shared between threads in a block and cannot be shared between blocks. Read/Write must be synchronized with syncthreads().Copying the global to shared for best use (but only if you can reduce global memory usage).Global Memory Variables declared with __device__ are stored in global memory, which is very slow. Lots of global memory, though (up to 2GB on our GPU) cudaMemcpy reads and writes from the CPU to GPU memory and can't be synchronize across blocks. For the synchronization process the multiple kernel invocations must be needed. Global Memory is declared on the host process using cudaMalloc and freed in the host process using cudaFree. Pointers are passed from the CPU to the GPU. Reducing global accesses is a goal, but an art form. Judicious use of shared memory is helpful. Constant Memory Variables that are declared with the __constant__ attribute are declared in constant memory (which is part of global memory). There is only a limited amount of constant memory (64KB per kernel), but it is much faster than regular global memory, because it is always cached. Constant memory can be written to by the host process using the cudaMemcpy To Symbol function and read-from using the cudaMemcpy from symbol function but can only be changed by the CPU.

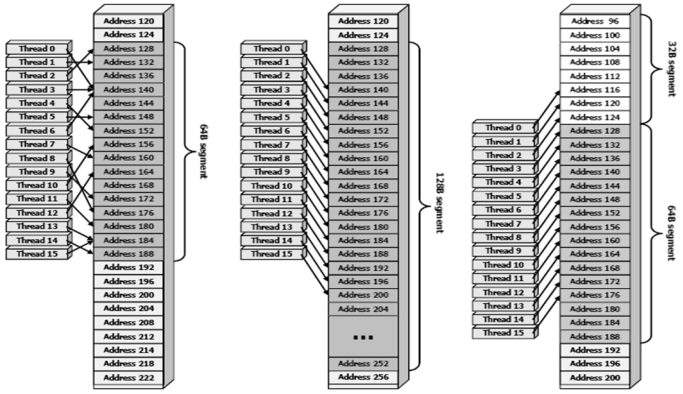


Figure 4.Memory operation in CUDA Architecture

©IJRASET: All Rights are Reserved

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue V, May 2018- Available at www.ijraset.com

The matrix multiplication done by breaking into grids. Each thread block defines a pair of shared memory buffers that are used to "cache" a "tile" of data from matrix A and matrix B. Since the "tile" is the same size as the thread block, we can just let each thread in the thread block load a single element from matrix A into one of the shared memory buffers and a single element from matrix B into the other as in figure 4.

IV. NUMBER THEORETIC TRANSFORM

A mathematical framework for Number theoretic transform is based on the theory of congruences modulo M, which belongs to the general area of what is often called "number theory." Number theory is very old, going back several thousand years. In recent years, there has been increasing interest in the practical applications of various parts of number theory, including the theory of residue number systems. There has been some work on the use of these number systems in general purpose although this line of investigation has not yielded many practical results due to the difficulty of determining the sign of numbers expressed in residue number system notation. More promising results have been obtained in applications where sign detection is not required, such as number theoretic transforms can be used to compute convolutions approximately three times as fast as the FFT implementation for the same convolution.

A. Numerical Expressions of Double Modulus Number Theoretic Transform

Fast implementation of convolution, and discrete Fourier transform (DFT) computations, are frequent problems in signal and image processing. In practice these operations are most often implemented using fast Fourier transform algorithms. NTTs can, in some instances, outperform FFT-based systems. In addition, it is also possible to use a rectangular transform, like the Walsh–Hadamard or the arithmetic Fourier transform, to get an approximation of the DFT or convolution and the NTT is defined over a finite group, as the transform pair,

$$X(k) = \sum_{n=0}^{N-1} x(n)\alpha^{nk} \bmod p,\ k=0,1,2....N-1$$
$$x(n) = N^{-1} \sum_{k=0}^{N-1} X(K)\,\alpha^{-nk} \bmod p, n=0,1,2.N-1 \quad (1)$$

Number Theoretic Transforms (NTT), made their appearance and with an aim of replacing the FFT as a tool of signal processing. These transformations based on rules of the number theory. However, in spite of the advantages of this transformation (exactness of the results), the NTT did not know a success similar to that of the FFT because of the interpretation problem of the field that it create, which is not the case for the frequency field created by the FFT. This transformation bases on the modulo-M properties. Hence, the good choice of M may gives properties similar to those of the Discrete Fourier Transform (DFT).Number Theoretic Transform (NTT) is a specialization of Fast Fourier Transformation (FFT), which complex functions are transformed by modulo operation makes the transform simple. The NTT is a generalization of the classic DFT to finite fields. With a lot of work, it basically lets one perform fast convolutions on integer sequences without any round-off errors, guaranteed. Convolutions are useful for multiplying large numbers or long polynomials and the NTT is asymptotically faster than other methods like Karatsuba multiplication.

B. Algorithm For Proposed Double Modulus Number Theoretic Transform

Input Vector: $X = (a_1, a_2, a_3 \ldots . a_n)$
1.Minimum Working Module :M ($1 \le n \le M$) in range [0,M]
2.Select K : $N=Kn+1=M$, $N \geq M$
3.Find the multiplication generator Z_M, $\omega = g^K$
α^((N-1)/K) ≡ 1 mod M
3.Generate Z_M $\alpha^{\frac{N-1}{K}} = 1 \bmod M$, $\alpha^{\frac{N-1}{K}} = 1 \bmod M$
$\omega = g^k, \omega = \alpha^k \bmod M$(primitive n^{th} root of unity)
4.Butterfly Structure
for ($0 \le j < K$)
if ($P \le M + 1$)
$P=X+Y$

©IJRASET: All Rights are Reserved

m=X+(~Y+1)
sx=(X×CX)+(Y×CY)
sy=(X×CY)+(~Y+CX+1)
end
end for.

C. Procedure For The Double Modulus Number Theoretic Transform

1) Consider the input vector is a sequence of n non-negative integers.
2) Choose a minimum working modulus M such that $1 \leq n < M$ and every input value is in the range [0,M]
3) Select some integer $k \geq 1$ and define N=kn+1 as the working modulus. We require $N \geq M$, and N to be a prime number. Dirichlet's theorem guarantees that for any n and M, there exists some choice of k to make N be prime
4) N is prime, the multiplicative group of Z_N of size

$\alpha(N)=N-1=kn$. the group must have a tleast one generator g, which is also a primitive $(N-1)^{th}$ root of unity

e) $\omega = g^k \bmod N$.then $\omega^n=g^{kn}=g^{N-1}=g^{\alpha(N)}=1 \bmod N$ due to Euler's theorem. g is a generator

$\omega^i = g^{ik} \neq 1$ for $1 \leq i \leq n$, because $ik < nk = N-1$. ω is a primitive root of unity as required by the DFT of length n

f) The rest of the procedure for the forward and inverse transforms is identical to the complex DFT. Moreover, the NTT can be modified to implement a Fast Fourier Transform algorithm such as Cooley-Turkey.The proposed system provides an additional modulus operation which will lead to elimination of complex number by taking modulus value for the twiddle factor hence the imaginary part becomes real part hence there is no wastage of memory buffers to store real and imaginary value separately. This will provide a revolutionary design to pay the way for multipliered architecture rather than the multiplierless architecture

V. RESULTS AND DISCUSSION

The Double Modulus Number Theoretic Transform has been computed and their results are measured for CPU-GPU operation which is simulated using the CUDA parallel architecture in Nvidia Graphic Cards using supporting language OpenCL, The specifications for the Graphics cards are given in the figure 5.

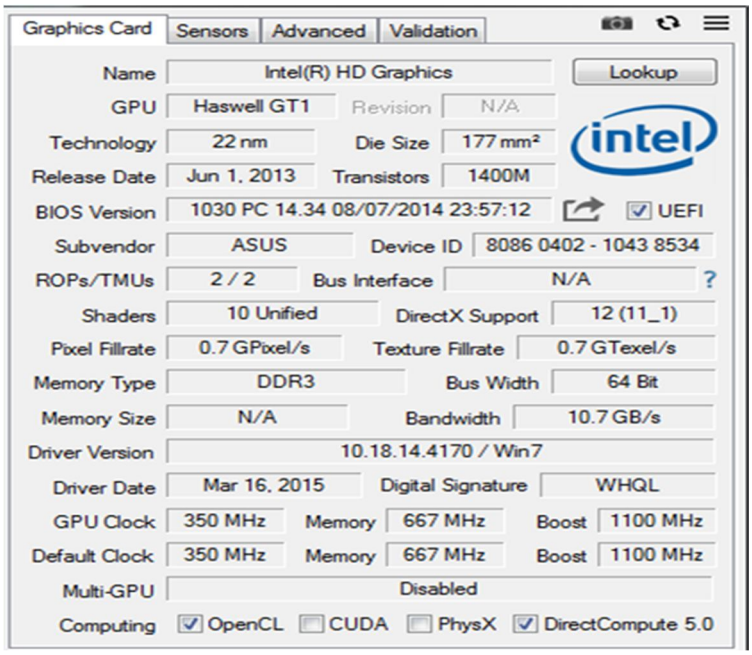


Figure 5.Technical Specifications of Graphic Card.

The results simulated are using Nvidia Graphic Cards with GPU clock time 350MHz and GPU memory of 667 MHz using the OpenCL computing. The Double Modulus NTT simulated using the CUDA parallel architecture in Nvidia Graphic cards using the language OpenCL the proposed architecture has been developed and the time taken for memory coping operation from CPU to GPU and utilization time taken to copy from GPU to CPU are measured. Their GPU utilization time for whole matrix multiplication has

©IJRASET: All Rights are Reserved

been measured are shown in figure 6 and time required by the memory computation time is less than that of the main matrix multiplication operation than the conventional methods available in GPU operations.

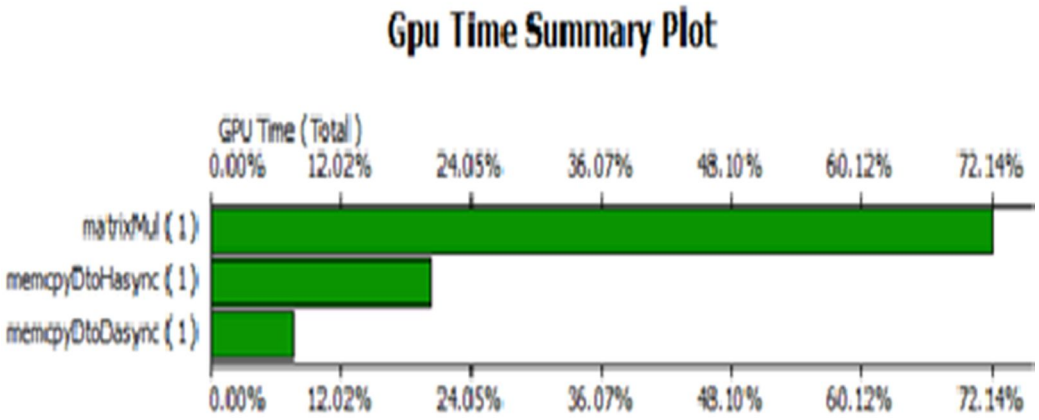


Figure 6.GPU Utilization plot for Double Modulus Number Theoretic Transform

The number of points in FFT and Double Modulus NTT has been compared with execution time in milliseconds with reference values with [9] and Double Modulus NTT provides less execution time than other existing system taken as reference shown in figure 7

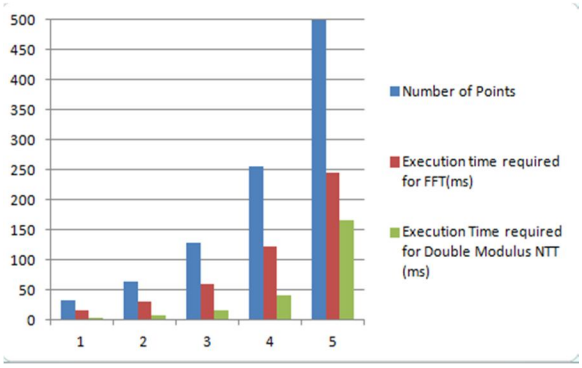


Figure 7. A graph representing the number of points compared to FFT and Double Modulus NTT execution time

From the above results the Double Modulus NTT can be efficient both in area and energy utilization. Hence the proposed architecture finds good for large number of input sizes and large data computation through which the communication between the multicores in GPU can also be speed up with less execution time

V. CONCLUSION

The proposed double modulus Number theoretic Transform (NTT) method enlarges the digit size of NTT and the double modulus technique provides a modulo operation which converts the imaginary data input into a real number which leads to reduction in buffer storage. The double modulus NTT technique is computed and simulated using the CUDA parallel architecture in Nvidia Graphic cards. Hence area occupied by the buffer can be greatly reduced and which makes an area efficient multiplication. By doing so this design can pay way for the multipliered architecture rather than going for multiplier less architecture and this idea of concept has been implemented in CUDA parallel memory architecture in Nvidia graphics processor using OpenCL computing which is an area and energy efficient architecture for memory operation in Graphic cards which provides a matrix multiplication between the threads and address of the memory blocks which can be employed in heterogenous processing core.By the Double Modulus NTT parallel code allows multi-core chips to give higher performance at lower energy.This can be a big factor in mobile devices that operate on batteries. Since each core in a multi-core CPU is generally more energy-efficient, the chip becomes more efficient than having a single large monolithic core. This allows higher performance with less energy and the Number theoretic Transform can also be implemented in various fields in cryptography for secure signal processing and in high efficient multiplication algorithm like Karatsuba multiplication algorithms.

©IJRASET: All Rights are Reserved

REFERENCES

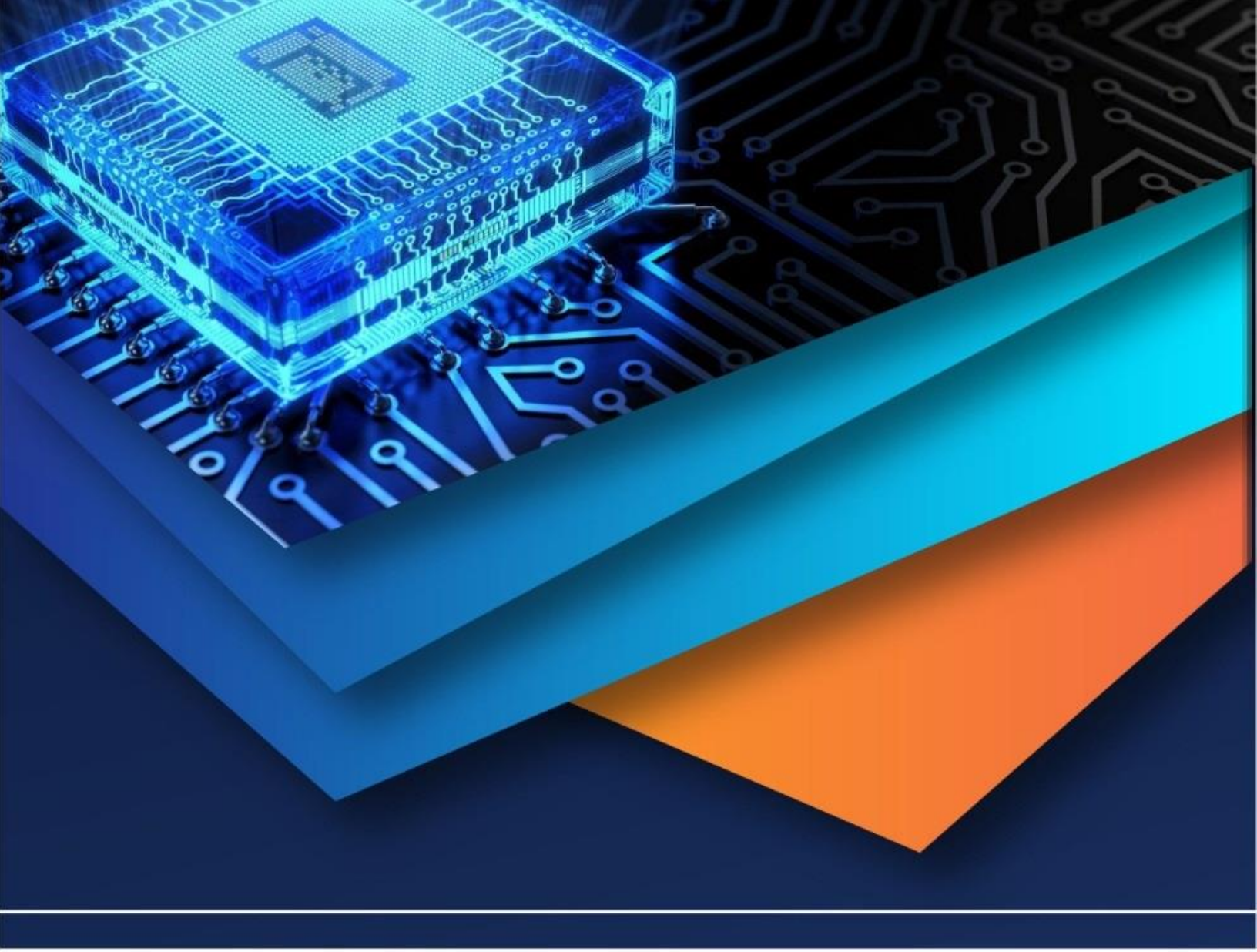
[1] Bartosz Pikacz; Jacek Gambrych; "Vector implementation of the Fast Fourier transform on DSP and NVIDIA CUDA platforms"2014 10th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME) Year: 2014
[2] Kyai Shah Dr.S.N. Pradhan "Performance & Limitations of CUDA on GPU for Computation of Fourier Transform" Conference: National Conference on Information Scieces (NCIS-2010)
[3] Muhamad Ibrahim;Omar Khan" Performance analysis of Fast Fourier Transform on Field Programmable Gate Arrays and Graphic crards" 2016 International Conference on Computing,Electronic and Eleactrical Engineering(ICE Cube) year:2016.
[4] Jiaxi Hu; Zhaosen Wang; Qiyuan Qiu; Weijun Xiao; "Sparse Fast Fourier Transform on GPUs and Multi core CPUs"2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing Year: 2012.
[5] Ulloa; Juan RamonTroncoso Pastoriza; Fernando PerezGonzalez"Number Theoretic Transforms for Secure Signal Processing"IEEE Transactions on Information Forensics and Security Year: 2017, Volume: 12, Issue: 5.
[6] Gavin Xiaoxu Yao;C.C.Cheung XiaoxuYao;CetinKaya Koc; "Reconfigurable Number Theoretic Transform architectures for cryptographic applications" 2010 International Conference on Field-Programmable Technology Year: 2010.
[7] Juliano B. Lima; Ricardo M. Campello de Souza; Paulo Hugo E. S. Lima" Fracttional number-theoretic transforms based on matrix functions"2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) Year: 2014.
[8] O. M. Ulyanov; M. S. Plakhov; A. I. Shevtsova; A. O. Skoryk; V. N. Tkachev; O. O. Ulyanova "Testing the speed of FFT/IFFT using the NVIDIA graphics cards"2016 9th International Kharkiv Symposium on Physics and Engineering of Microwaves, Millimeter and Submillimeter Waves (MSMW) Year: 2016.
[9] Xiang Feng; Shuguo Li "Design of an area efficient Million bit Integer Multiplier Using Double Modulus NTT" IEEE Transactions on Very Large Scale Integration (VLSI) SystemsYear: 2017, Volume: 25, Issue: 9 Pages: 2658 - 2662 IEEE Journals & Magazines.
[10] Boon-Chiao Chang; Bok-Min Goi; Raphael C. -W. Phan; Wai-Kong Lee"Accelerating Multiple Precision Multiplication in GPU with KeplerArchitecture" 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)

AUTHORS

C.Sai Punitha
Post Graduate Student,
Department of Electronics and Communication Engineering,
Easwari Engineering College,Chennai,India.
Email: punithacs24@gmail.com

D.Jessintha
Associate professor,
Department of Electronics and Communication Engineering,
Easwari Engineering College,Chennai,India.
Email: jessintha.kumar@gmail.com

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)