ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Analysis of 'HeapBitonic Sort'Under Different GPU Architectures

Kapil Kumawat[1], Rohit Maheshwari[2], B.L Pal[3]

M.Tech Scholar, Dept. of Computer Science & Engineering, Mewar University, Rajasthan, India[1]

Asst. Professor, Dept. of Computer Science & Engineering, Mewar University, Rajasthan, India [2]

Asst. Professor, Dept. of Computer Science & Engineering, Mewar University, Rajasthan, India [3]

ABSTRACT - Implementation and compared the sorting time of HeapBitonic sort under different GPU architectures with 16 bit float and 16 bit integer types of data set. For that , Radeon (TM) HD 8670M GPU and NVIDIA GeForce GT 520MX GPU are used on different systems. The Radeon (TM) HD 8670M GPU based on GCN architecture and NVIDIA's GeForce GT 520MX GPU based on Fermi architecture. The obtained results revealed that the Fermi architecture of GPU is the best way to obtain algorithmic efficiency than GCN architecture because of it has more streaming multiprocessor cores and uses the fast on chip memory of GPU.

KEYWORDS - CUDA, OpenCL , threads , pipelines ,GPU

I. INTRODUCTION

Now a day's various sorting techniques are implemented on parallel machines so that parallel architectures can be used to increase the performance of a system are the development of multi core processors. In multi core processors, two or more processors are used in order to rise efficiency and enhance performance. Recently, the graphics processing units (GPU) based on multiple processing cores represent feasible solutions for increasing the level of parallelism.

A multi core GPU based sorting algorithm achieves high performance in sorting large scale data by exploiting parallel core processors. Multi core processors are essential to compute deep parallel computations and they could be used to assist the GPU in solving problems that can be parallelized efficiently. It has been exploited in many different parallel general purpose applications. Multi core GPU is suitable for highly parallel process because of higher memory bandwidth, thousands of hardware threads contexts with hundreds of parallel programs in a SIMD fashions.

OpenCL(Open Computing Language) [9] and CUDA [17] are the programming models for multi core processors. OpenCL framework is used to make parallel programs that execute on central processing units (CPU's) and graphics processing units (GPU's). OpenCL offers parallel computing using task-based and data-based parallelism [9]. OpenCL is emerging as a standard for heterogeneous multicore GPU systems. It allows the same code to be executed across on different processors like multi core CPUs and GPUs. OpenCL offers a single programming framework, which can be used to target multiple platforms from different vendors. The OpenCL applications provides a various OpenCL extensions and optional features that are designed to utilize all available resources on CPU's and GPU's [2].

II. LITERATURE REVIEW

Mohammad H. Al Shayeji et. al. , 2014 proposed "Hybrid Multi-Phased GPU sorting algorithm (HMP) - HeapBitonic" that exploits the parallelism of modern GPU architecture. Their evaluation and discussions shows that the proposed algorithm has less execution time when compared to bitonic-sort, merge-sort and even-odd sort under different types of datasets [1].

Pirjan ,2011 describes "Optimizing Techniques for Data Sorting Algorithms" and design parallel radix sort for many cores graphics processing units, benefitting from the high computational power offered by the Compute Unified Device Architecture (CUDA).In order to optimize the radix sort, he divided the sequence into tiles that have been assigned to a number of thread blocks [2].

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Nadathur Satish et. al.,2014 designed "Efficient Sorting algorithms for many core GPUs". They describe the design of high performance parallel radix sort and merge sort routines for many core GPUs, taking advantage of the full programmability offered by CUDA. Their radix sort is the fastest GPU sort and our merge sort is the fastest comparison based sort reported in the literature. They carefully designs their algorithms to expose substantial fine-grained parallelism and decompose the computation into independent tasks that perform minimal global communication. They exploit the high-speed on chip shared memory provided by NVIDIA's GPU architecture and efficient data-parallel primitives particularly parallel scan [3].

Mr. Sunil Kumar Pandey et.al.,2013 proposed "The Efficient load balancing in the parallel Computer" which is based on the efficient load balance feature in the parallel Computer .The mathematical analysis of their study identify various useful results. If all the parallel computer are not same type means not same configuration then proper load balance not occur so some computer finish their work earlier than other and sit ideal which degrade the performance of multicomputer system. They introduce new algorithm for program development on multicomputer environment called ODDA (optimize data distribution algorithm) to address the proper load balancing purpose. This algorithm works in the master slave modal. The main processor called the master processor which dynamically assigns the work load to the remaining cooperative slave processor in run time environment [4].

Krishnahari Thouti et. al. ,2012 studied " Comparison of OpenMP &OpenCL Parallel Processing Technologies". They presents a comparison of OpenMP and OpenCL based on the parallel implementation of algorithms from various fields of computer applications. The focus of their study is on the performance of benchmark comparing OpenMP and OpenCL. They observed that OpenCL programming model is a good option for mapping threads on different processing cores. Balancing all available cores and allocating sufficient amount of work among all computing units, can lead to improved performance [5].

Mwaffaq A. Abu et. al. ,2008 presents "A Heapify Based Parallel Sorting Algorithm". Their key idea is to enhance the existing quick sort method. The algorithm consisted of several stages, in first stage; it splits input data into two partitions, next stages it did the same partitioning for primary stage which had been spitted until 2 m partitions was reached equal to the number of available processors, finally it used heap sort to sort respectively ordered of non internally sorted partitions in parallel. Their results showed the speed of algorithm about double speed of classical Quick sort for a large input size [6].

Nikolaj Leischner et.al.,2009 presents "GPU Sample sort". The design of sample sort algorithm for many core GPUs. Despite being one of the most efficient comparison-based sorting algorithms for distributed memory architectures. For uniformly distributed keys of sample sort is at least 25% and on average 68% faster than the best comparison-based sorting algorithm, GPU thrust merge sort, and on average more than 2 times faster than GPU quick sort. Moreover, for 64-bit integer keys it is at least 63% and on average 2 times faster than the highly optimized GPU thrust radix sort that directly manipulates the binary representation of keys. Their implementation is robust to different distributions and entropy levels of keys and scales almost linearly with the input size. These results indicate that multi-way techniques in general and sample sort in particular achieve substantially better performance than two-way merge sort and quick sort [8].

III. PROBLEM STATEMENT

Mohammad H. Al Shayeji et. al. , 2014 proposed a "Hybrid Multi-Phased GPU sorting algorithm - HeapBitonic Sort", a parallel algorithm for sorting large set of data on GPU [1]. But has some limitations , these are

a. The HeapBitonic sort does not test under different multi core GPU architecture.
b. The HeapBitonic sort does not use other data types such as float and double.
c. The HeapBitonic sort takes an additional time at lower data size is due to the overhead of partitioning data into sub-sequences and merging.

IV. OBJECTIVE

The main objective is to implement and analyzed the execution time of the existing 'HeapBitonic' sort using both 16-bit float and 16 bit integer types of data set with different GPU architectures (GCN & Fermi).

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

V. ARCHITECTURAL FRAMEWOK-OPENCL MEMORY MODEL

OpenCL is a framework for parallel programming and includes a language API, libraries and a runtime system to support software development and to access and control the devices[14].

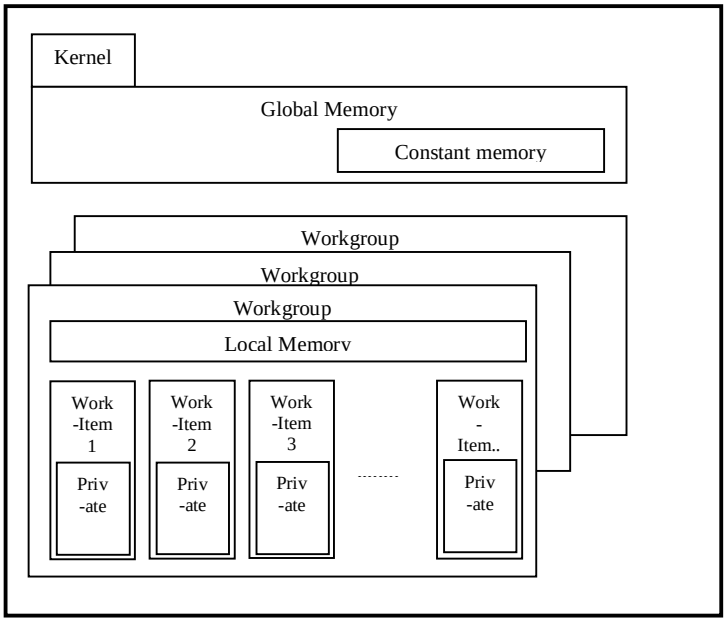

Figure 1. OpenCL Memory Model [15]

OpenCL is used for writing programs in parallel that execute across different platforms, most importantly both CPU's and GPU's. The OpenCL memory model consists hybrid model where multiple kernels each with multiple work items are enqueue for execution at the same time. It provides a region of private memory to a work-item in which the variables defined in one work item's private memory are not revealed to another work item. On the other side local memory is modelled as being shared by a workgroup. This memory region can be used to store variables that can be shared by all work-items in a work group [12].

The global memory is seen by all processing elements on the device (similar to the main memory on a CPU-based host system). Whenever the data is shifted from the host to the device, the data will reside in global memory. Any data that is to be reside back from the device to the host must also reside in global memory.

Constant memory : is not only designed for every type of read-only data but also for data where each element is accessed simultaneously by all work-items. Variables whose values never change also fall into this category. Constant memory is modelled as a part of global memory, so memory objects that are transferred to global memory can be specified as constant [12].

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

VI. COMPARISON OF TECHNICAL SPECIFICATION BETWEEN AMD RADEON HD 8670M & NVIDIA'S GEFORCE GT 520MX GPU

S.No	Parameters	Radeon (TM) HD8670M	NVIDIA GeForce GT 520MX
1	Manufacturer	AMD	NVIDIA
2	Technology	28 nm	40nm
3	Design architecture	GCN	Fermi
4	Die Size:	77 mm^2	79mm^2
6	DirectX	11.0/SM 5.0	11.0/SM 5.0
7	OpenGL:	4.2	4.4
8	OpenCL:	1.1	1.1
9	PCI e version	3.0 x16	2.0 x16
10	Memory Size	2048 MB	1024MB
11	Memory Type	DDR3	DDR3
12	memory bandwidth	12.8 GB/sec	14.4 GB/sec
13	Memory Bus width	64 Bit	64 Bit
14	GPU Clock:	350 MHz	900 MHz
15	Default Clock:	350 MHz	900 MHz
16	GPU Memory	800 MHz	900 MHz
17	Shading Units:	16	48
18	TMUs:	8	8
19	ROPs:	4	8
20	Pixel Fill Rate:	1.4 GPixel/s	1.8 GPixel/s
21	Texture Fill Rate:	2.8 GTexel/s	7.2 GTexel/s

Table 1: Comparison of technical specifications between AMD Radeon HD 8670 M (GCN) and NVIDIA GeForce GT 520MX (Fermi)

VII. EXPERIMENTAL SETUPS-HARDWARE & SOFTWARE REQUIREMENTS

All experiments are done using system consisting of Radeon(TM) HD 8670M and NVIDIA GeForce GT 520MX video graphic cards. The Radeon GPU supports PCIe ver 3.0. It has 300 MHz core speed and supports GCN architecture. The memory is 2048 MB with 2000MHz speed. The Memory type is DDR3 with 4.8 GB/sec memory bandwidth. The host machine used is HP Pavilion 15 Note Book PC, 15-inch, with 1.80 GHz Intel core i3 CPU and 4 GB DDR3 main memory.

On the other side, NVIDIA GeForce GT 520MX GPU is used for the same sets of data set sorts according to the method. The NVIDIA GPU supports PCIe 2.0 x16. It has 900 MHz core speed and supports Fermi architecture. The memory is 1024MB with 2000MHz speed. The Memory type is DDR3 with 14.4 GB/sec memory bandwidth. The host machine used is Lenovo Note Book PC, 15-inch, with 2.50 GHz Intel core i5 CPU and 4 GB DDR3 main memory.

The application is made in C language for that visual studio 2012 tool is used, SDK driver version OpenCL 1.1,CUDA toolkit version 7.5.18 and development drivers and Intel SDK for OpenCL application kernel Builder.

VIII. IMPLEMENTATION & PARAMETERS

The HeapBitonic sort[1] algorithm is analysed by using OpenCL programming platform using both 16 bit integer and 16 bit float types of data set and compare their sorting time results under different GPU architectures (GCN & Fermi).

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

IX. EXPERIMENTAL RESULTS & ANALYSIS

On implementing the HeapBitonic [1] sorting algorithm with 16 bit integer and 16 bit float types of data sets ,on systems with different GPU's having GCN and Fermi architectures .We analysed the sorting time results for 16 bit integer types of data set using GeForce GT 520MX GPU (Fermi) & Radeon HD 8670M GPU (GCN). The results of their sorting time are as follows .

		Graphics Processors	
S.No	**Array Size (N)**	**Radeon (TM) HD 8670M (Time in milli sec.)**	**NVIDIA GeForce GT 520MX (Time in milli sec.)**
1	512	.011	.002
2	1024	.011	.003
3	2048	.011	.003
4	4096	.014	.004
5	8192	.020	.006
6	16384	.031	.011
7	32768	.055	.024
8	65536	.050	.044
9	131072	.098	.086
10	1048576	.940	.089
11	2097152	1.896	.092

Table 2: Sorting for different input sizes (16 bit Integer) on Radeon(TM) HD 8670M GPU (GCN) and GeForce GT 520MX GPU (Fermi).

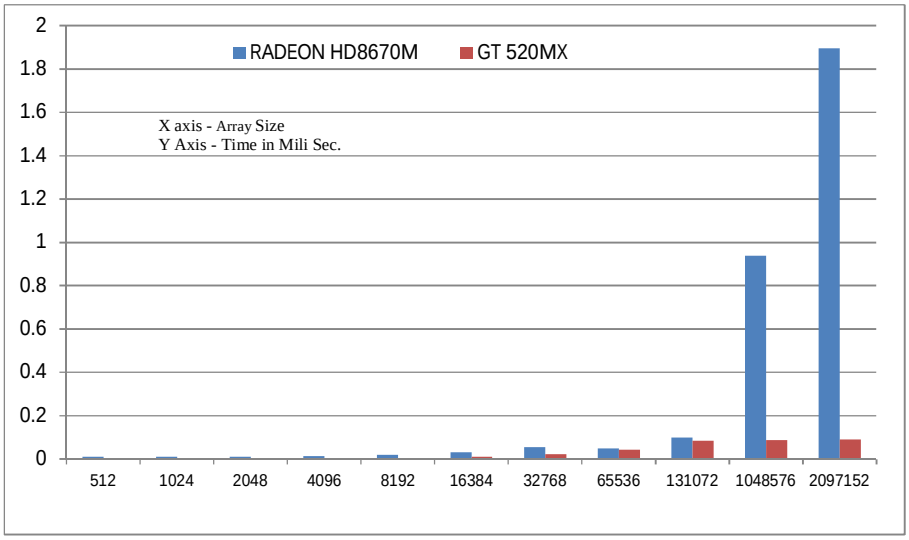

Figure 2. Comparative analysis between GeForce GT 520MX (Fermi) and Radeon HD 8670M (GCN) for 16 bit integer types of data set.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

It is to be noted that the time measured here is the run time or the profile time excluding the time for memory allocation, data and memory transfers between the host and the device. The time has been measured through the API clGetEventProflingInfo. In figure 2, It is clearly seen from the that GeForce GT 520MX GPU takes less execution time comparative to the Radeon (TM) HD 8670M GPU when we use them for 16 bit integer types of data set. For Both smaller and greater types of data set takes less execution time. For greater data set its clearly seen that the execution time remains same (approximately).

We analysed the sorting time results for 16 bit float types of data set using GeForce GT 520MX GPU (Fermi) & Radeon HD 8670M GPU(GCN). The results of their sorting time are as follows .

		Graphics Processors	
S.No	**Array Size (N)**	**Radeon (TM) HD 8670M (Time in milli sec.)**	**NVIDIA GeForce GT 520MX (Time in milli sec.)**
1	512	.011	.003
2	1024	.010	.003
3	2048	.011	.003
4	4096	.013	.004
5	8192	.019	.006
6	16384	.031	.011
7	32768	.055	.025
8	65536	.100	.047
9	131072	.206	.091
10	1048576	1.828	.093

Table 3: Sorting for different input sizes(16 bit float) on Radeon(TM) HD 8670M GPU (GCN) and GeForce GT 520MX GPU (Fermi).

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

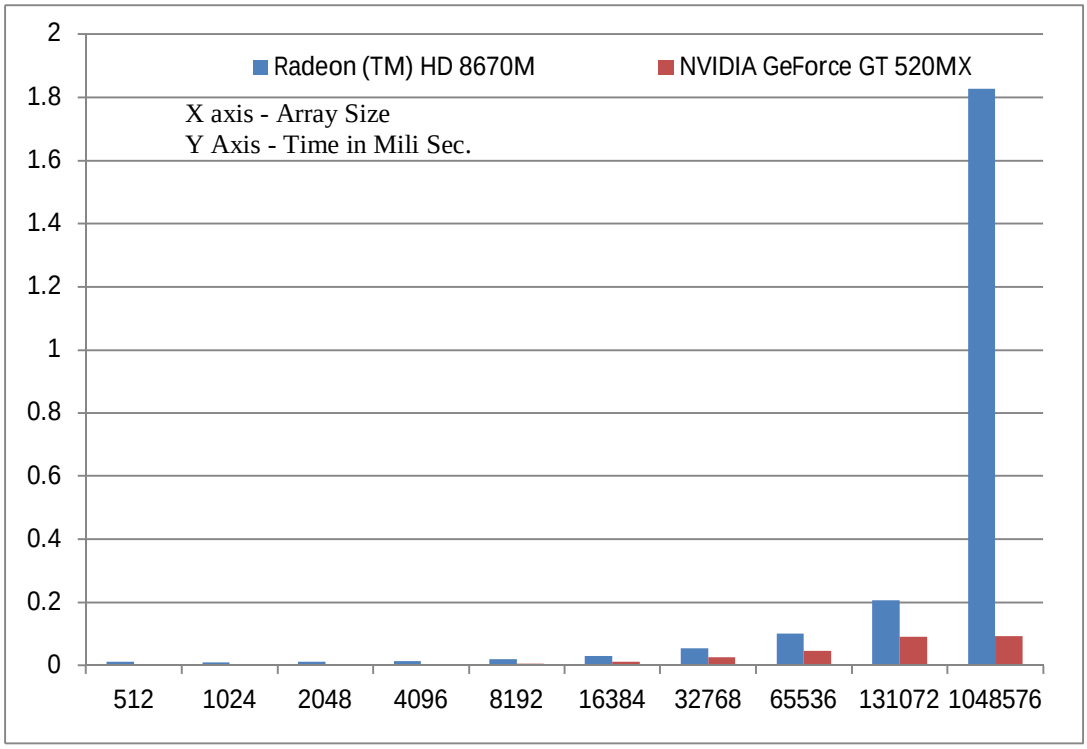

Figure 3. Comparative analysis between Nvidia GeForce GT 520MX (Fermi) & Radeon HD 8670M (GCN) for 16 bit float types of data set.

It is clearly seen from the figure 3,that GeForce GT 520MX GPU takes less execution time comparative to the Radeon (TM) HD 8670M GPU when we use them for 16 bit float types of data set. For Both smaller and greater data set takes less execution time due to the advantage of the Fermi architecture provides more streaming processing cores so that it gives significant improved execution time of sorting technique.
The obtained numerical results have confirmed and revealed that the best way to obtain algorithmic efficiency on Fermi architecture. It uses the fast on chip memory of GPU and to employ a fine grained parallelism in order to benefit from the computing power of thousands of parallel threads offered by Fermi architecture. Fast memory speed have a significant influence on the overall performance and must be properly managed by the GT 520MX GPU.

X. CONCLUSION

On implementing HeapBitonic sort under different GPU architecture such as Fermi and GCN , it can be ascertained that HeapBitonic sort provides the better performance on GeForce GT 520MX GPU which is based on Fermi than RadeonHD 8670M because of the Fermi architecture provides more streaming processing cores of the GT 520MX ,it enhances and improves the execution time of sorting technique.

XI. FUTURE WORK

We can test HeapBitonic algorithm on other different GPU architectures such as kepler,Tesla,Maxwell with different data type like 32 bit integer and we can also test HeapBitonic algorithm on highly configured CPU using different data sets like 32 bit float and 32 bit double.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

REFERENCES

1. Mohammad H. Al Shayeji, Mohammed H Ali and M.D. Samrajesh,“ A Hybrid Multi-Phased GPU Sorting Algorithm”, Computer Engineering Department, College of Computing Sciences and Engineering, Kuwait University, Kuwait , 8(23) , 315-322,2014.
2. Pirjan ,” Optimization Techniques For Data Sorting Algorithms”, Ann. of DAAAM & Proceedings of the 22nd International DAAAM Symp., Vol 22, No. 1, 1726-9679, 2011.
3. Nadathur Satish, Mark Harris and Michael Garland,” Designing Efficient Sorting Algorithms for Many core GPUs”,2014.
4. The Efficient load balancing in the parallel Computer Mr. Sunil Kumar Pandey1, Prof. Rajesh Tiwari2 Computer Science and Engineering Department, shri sankaracharya College of Engineering & Technology Bhilai, International Journal of Advanced Research in Computer and Communication Engineering, ISSN (Print) : 2319-5940 ISSN (Online) : 2278-1021, Vol. 2, Issue 4, April 2013.
5. Krishnahari Thouti, S.R.Sathe ,”Comparison of OpenMP & OpenCL Parallel Processing “ Technologies (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.4, 2012.
6. Mwaffaq A. Abu Al hija, Arwa Zabian, Sami Qawasmeh and Omer H. Abu Al haija,” A Heapify Based Parallel Sorting Algorithm”, Journal of Computer Science 4 (11): 897-902, 2008.
7. Khronos OpenCL Working Group. The OpenCL Specification, version 1.0.29, 8 December 2008.
8. Nikolaj Leischner,Vitaly Osipov, Peter Sanders,30 sep 2009,” GPU Sample Sort”.
9. OpenCL Framework and application::: http://www.khronos.org/opencl
10. Cheng ZHONG, Wei WEI, Mengxiao YIN, Yiran HUANG ,” Cache-efficient Parallel Multi-way Merging for multicore machines". Journal of Computational Information Systems 10: 2 (2014) 789–797, 2014.
11. GCNArchitecture:http://notebookcheck.net/ATIRadeonHD3200.9591.html.
12. Benedict R.Gaster,Lee Howes,david kaeli,Perhaad Mistry,Dana Schaa,”Heterogeneous computing with OpenCl” 2012.
13. The opencl specification ,version 2.0 Khronos opencl working group , Revised date 17 oct 2014.
14. Khronos Group. (n.d.). OpenCL Overview: OpenCL—The Open Standard for Parallel Programming of Heterogeneous Systems. https://www.khronos.org/opencl. Retrieved March 2011.
15. Introduction opencl programming training guide ,publication 137-41768-10,Rev-A,Issue Date May 2010.
16. OpenCL Parallel Programming Development Cookbook,Raymond Tay,ISBN 978-1-84969-425-0.
17. NVIDIA OpenCL Programming Guide for CUDA Architecture, Version 3.2, VOL 6, No. 2 (March/April 2008).
18. Dominik Grewe and Michael F.P. O'Boyle, "A Static Task Partitioning Approach for Heterogeneous Systems Using OpenCL". School of Informatics, The University of Edinburgh, UK.
19. V. Mansotra and Kr. Sourabh,March 10 – 11,"Implementing Bubble Sort Using a New Approach",Proceedings of the 5th National Conference; INDIACom-2011, 2011.
20. Atanas Radenski ,"Shared Memory, Message Passing, and Hybrid Merge Sorts for Standalone and Clustered SMPs", School of Computational Sciences, Chapman University, Orange, California, USA, the 2011 internation conference on parallel and Distributed Processing Techniques and Applications ,CSERA Press ,2011,pp.367-373, 2011.
21. Nvidia corporation, nvidia's next generation cuda compute architecture : Fermi::.http://www.nvidia.in/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf, 2009.
22. NVIDIA GPU Architectur https://en.wikipedia.org/wiki/Fermi_(microarchitecture
23. David A. Patterson and John L. Hennessy. Computer Organization and Design. Morgan Kaufmann, 2009.
24. Nvidia and AMD generation GPU Performance and Hardware setups : http://www.geforce.com/hardware/notebook-gpus/geforce-gmx/performance.
25. Nvidia corporation, nvidia's next generation cuda compute architecture : http://www.nvidia.in/object/product-geforce-gt-520mx-in.html.
26. NVIDIA-GeForce-GT 20MX techpower GPU Database: http://www.notebookcheck.net/NVIDIA-GeForce-GT 20MX.54717.0.html.
27. GCN White paper, amd graphics cores next (gcn) architecture.:http://www.amd.com/Documents/GCN_Architecture_whitepaper.pdf

BIOGRAPHY

Kapil Kumawat, is an M.tech scholar in the department of science & engineering ,Mewar University , Chittorgarh , Rajasthan. My research interests is in Parallel Computing ,Data Structures and Algorithms.