Fuzzy Systems using GPGPU – A Survey

Satvir Singh[1] and Shivani Kakkar[2]
[1,2]*Department of Electronics & Communication Engineering,*
Shaheed Bhagat Singh State Technical Campus Moga Road, Ferozepur–152004, Punjab, India
E-mail: [1]drsatvir.in@gmail.com, [2]kakkarshivani47@yahoo.in

***Abstract*—This paper presents a survey on use GPGPU (General Purpose computing on Graphics Processing Unit) to implement Fuzzy Logic Systems (FLSs). Features such as massively parallel, multithreaded operations, many-core processor make Graphics Processing Unit (GPU) suitable for real-time applications. Inherent parallel nature of Type-1 and Type-2 FLSs has been exploited for parallelization on GPU. Various applications, like fuzzy clustering, image processing, robot navigation, and fuzzy arithmetic library, etc. have been studied for better performances on GPU using CUDA (Compute Unified Design Architecture) programming model. In present scenario of High Performance Computing (HPC), GPGPU is most significant low cost solution for many engineering problems.**

Keywords: GPU, GPGPU, CUDA, Type-1 and Type-2 FLSs

I. Introduction

Graphic Processing Unit (GPU) developed in 1970s is traditionally used for texture and video rendering. GPU is exceptionally suited for HPC just because of its large number of computational cores. The high speed processors inside GPU have 100s of ALUs running 1000s of identical threads in parallel to execute instructions simultaneously. Tasks performing identical operations which are independent of each other execute on many data elements in parallel. CPU spends a lot of time on computations as compared to GPU. So, the GPU is faster as compared to CPU owing to larger number of computations being performed on processing cores of GPU simultaneously. GPU being classically used for texture and graphics applications is now playing a vital role in speed up of many computationally intensive algorithms. This general-purpose parallel computational functionality of GPU is supported by the scalable CUDA programming model [16][18]. In 2007, first release of CUDA explored the parallel architecture of GPUs for parallel computing for various applications other then graphics. It enables GPU to execute programs written in C. It is a small set of extensions to C/C^{++} and enable heterogeneous programming including provisions for both host (CPU) and device (GPU) through PCI express bus. Data parallel portions of an algorithm are executed on the device as kernels. One kernel is executed at a time by many threads in a block. CUDA threads on a GPU can be executed independently and each thread performs the same operation and execute same kernel as shown in Fig. 1, from architectural point of view.

CUDA uses software and hardware required for making GPU hardware easily accessible to programmers. Not only CUDA, there are many other programming platforms, like Shader, OpenCL, and open GL, etc. for exploiting GPGPU. In CUDA, to start application data is copied from CPU to GPU memory over a PCI Express bus followed by load and execute program. Then program results are copied back from GPU memory to CPU memory.

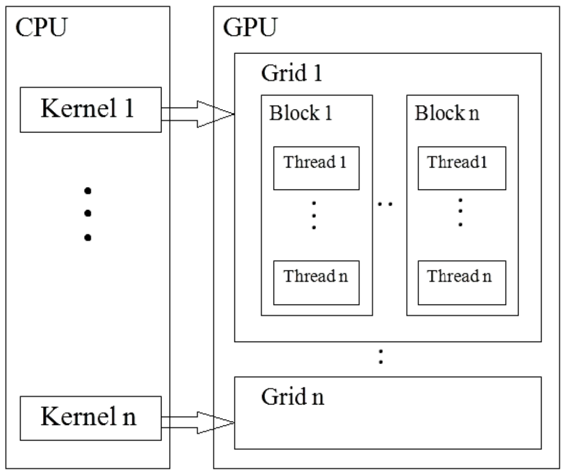

Fig. 1 CUDA processing Model Design

As FLSs possess inherent parallel nature [1], it is easy to exploit them on parallel architecture of GPUs for implementation. This paper presents various Fuzzy Logic engineering problems where GPU has been investigated for increased speedups. Organization of rest of this paper as: Section I presents an overview of Type-1 and Type-2 FLSs. Section II discusses about basics of Type-1 and Type-2 FLS and Section III reports various GPGPU implementations for FLS concepts and applications. Finally, Section IV summarizes the paper and presents motivational offshoots for the HPC researchers working on FLSs.

II. Fuzzy Logic Systems

The term Fuzzy Logic was introduced by Lotfi A. Zadeh in 1965 [2][3]. In 1970s, research groups formed in around the world investigated fuzzy logic where conventional mathematical tools face difficulties in handling engineering (especially, control) problems. Fuzzy sets provide provision for dealing with vagueness and ambiguity. In fuzzy sets, each element is mapped within [0, 1] by an analog membership function [4]. Rulebase is extracted from experiential fuzzy knowledge of experts to control the output variable. A fuzzy rule is a simple IF-THEN rule with a condition and a conclusion. For example, if *temperature* (input variable) is *cold* (fuzzy set) then output command is *heat* (fuzzy set). Aggregated fired fuzzy rules are subjected to defuzzification process to obtain a crisp output as resultant. The *max* operator and *Center of Gravity* are most preferred methods for aggregation and defuzzification, respectively.

A. Type-1 Fuzzy systems.

Type-1 fuzzy systems consist of inputs fuzzified using fuzzy sets, expert knowledge extracted in the form of fuzzy rulebase, inference engine, and defuzzifier as shown in Fig. 2. Type-1 fuzzy sets are incapable of handling uncertainties over uncertainties, i.e., second ordered uncertainty. So keeping in mind another type of fuzzy sets were introduced by Zadeh known as Type-2 fuzzy sets [3].

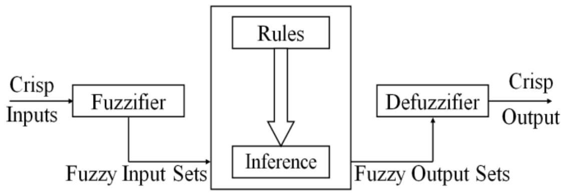

Fig. 2 Block Diagram Representation of an FLS [2]

B. Type-2 Sets & Fuzzy Systems

Fuzzy sets models words that are being used in rulebase and inference engine. However, word mean different thing to different people and, therefore, are uncertain. Membership degree of a Type-1 fuzzy set cannot capture uncertainties about the words. Hence, another type of fuzzy set, i.e., Type-2 fuzzy Sets, came into existence which is capable of handling such uncertainties. For such a fuzzy set membership value corresponding to some crisp input is not a crisp value rather a Type-1 fuzzy set called secondary membership [6][17]. This concept can be extended to Type-*n* fuzzy sets. Computations based on Type-2 fuzzy sets are very intensive, however, when secondary membership is assumed unity the computational burden reduces drastically. This is another variant to fuzzy set representation and is known as Interval Type 2 fuzzy sets [5][16][17].

III. FUZZY SYSTEMS USING GPGPU

C. Type-1 FLS

1) Fuzzy Inference System (FIS) on GPU

Here GPU is reviewed for speedup up of FLSs which is one of the non-graphics based applications. Derek T. Anderson, *et al*. along with his team investigated this by exploiting inherent parallel nature of FLSs. 128 processing units were operated in parallel thus making intense calculations of constructing rulebase and inference process faster as compared to that of CPU [7]. The GPU used was NVIDIA's Geforce 8800 GTX, having 128 stream processors, a core clock of 575MHz, shader clock of 1350MHz and that is capable of handling 350GFLOPs. GPU implementation has found 2 orders of magnitude faster as compared to CPU.

2) Mamdani FIS

Derek Anderson, *et al*. here exploited the HPC power of GPU to speedup the inference process inside Mamdani FIS [13]. Various steps of FIS, i.e. fuzzification, implication, aggregation and defuzzification are executed as separate CUDA kernels on GPU. NVIDIA 8800 BFGGTX GPU with 768 MB of texture memory was used. PCI express X16 was used. Number of inputs are kept as 2 whereas number of rules are varied as 16, 32, 64 and 128. In addition, discretization levels are varied as 256, 512, 1024, 2048 and 4096. Comparative analysis of CPU versus GPU is conducted for a series of 30 runs. Speedup of approximately a factor of 178 was obtained on GPU as compared to CPU. Parallelization of larger number of FISs and extension of same work to Type-2 fuzzy sets may be treated as an offshoot.

3) Fuzzy TSK tuning

Artificial Intelligence (AI) techniques are too slow to be computed on CPU in real-time. In 2012, Ferreira and Cruz have introduced special approach to offload parts of the AI computations, i.e., automatic training of fuzzy TSK tuning, of a game on to a GPU [8]. In TSK systems consequents for an output which are N-order polynomials are tuned using Batch Least Square (BLS) method and input fuzzy sets are tuned using gradient method. Both these methods of tuning are operated in parallel using CUDA. Gaussian membership function being continuous and easily differentiable is used in 2-input and 1-output FLS. In this MISO system, first input has five fuzzy sets whereas second has seven fuzzy sets and, hence, maximum thirty five fuzzy rules. Experiments are run on three different machines, (1) Geforce GTX 550 Ti, (2) Tesla C2070 and (3) Geforce GTX 590. In all these cases, GPU implementation surpassed CPU by five to six times. The method can be implemented for real-time applications, like games to learn the player's behavior and its adaptation to various circumstances over time. The purposed method can also be experimented for complex training patterns containing high dimensional inputs and number of rules in future.

4) Fuzzy arithmetic library on GPU

Fuzzy arithmetic library is introduced by David and Marin as solution to the problems which deals with the uncertainty and complex data representation in the form of integer and floating point [9]. Here with the use of CUDA based GPGPU execution time for basic operations (addition and multiplication) has been improved tremendously. All the techniques have been implemented using NVIDIA's GPU based on fuzzy numbers. The method used for implemented was midpoint-radius encoding and was compared with traditional lower upper encoding. Gain of 2 to 20 was obtained by preferring the former method over later. Evaluation of the accuracy of the new representation format is the extension of this work.

D. Fuzzy Logic Based Image Processing

The real time image processing using simple algorithm is computationally intensive task even with the moderate size images. With further increase in image size it becomes really a difficult task. Anderson *et al.* introduced parallelization of fuzzy logic based image processing where edge computation for each pixel being independent of all other pixels calculation is made parallel. GPGPU implementation using CUDA consisted of two CUDA kernels, one for rule firing and another for defuzzification [10]. The CPU and GPU implementations were then run over a series of different image sizes. Maximum of 126 times speed improvement to the original algorithm is achieved on a NVIDIA 8800 Ultra GPU, and hence making the processing of the algorithm real time. The most significant advantages of GPGPU implementation include its low price and ease of learning & using CUDA API. Moreover, such a high speed allows spending more time at higher level image processing operations, e.g., object recognition or tracking, etc. Various higher level processing operations can also be performed on GPU in future using a generalized GPU Mamdani FIS implementation.

Nowadays, with the quantitative increase in the research and practice of clinical radiology and also with the increased size of images, radiology to become practical in real time it is important to implement the image segmentation rapidly which is made possible by this paper. The Iterative Relative Fuzzy Connectedness (IRFC) segmentation is one of the families of fuzzy connectedness algorithm [11]. In order to segment large medical image data sets a parallel (IRFC) algorithm via image foresting transform is developed and implemented using NVIDIA's CUDA on GPU. The two major parts of the algorithm, (1) computation of fuzzy affinity relations and (2) then computing the fuzzy connectedness relations and tracking labels for objects of interest are computed as two separate CUDA kernels and a tremendous speed improvement could be achieved. The GPU used is Tesla C1060 GPU and speed increased by a factor ranging from 2.4 to 42.7 times. In future, automatic anatomy recognition in radiology can be easily implemented on GPU.

Fuzzy Anisotropic Diffusion (FAD) algorithm basically oriented for high resolution multidimensional image/video is considered to be computationally complex technique [12]. As fuzzy logic is inherently parallel in nature [1], FAD can be easily implemented in parallel on GPU using CUDA that replaces the recent methods for enhancement, reconstruction, post processing and classification procedure which are not feasible for real time implementation. The experiments are performed on both NVIDIA Tesla C2075 GPU using CUDA and on quad-core Intel Xeon E5603 CPU in the MATLAB environment. The implementation of FAD algorithm using GPGPU is found to be less time consuming, i.e., 140 times faster than that of MATLAB implementation on CPU. GPGPU implementation has also enhanced the resolution of the image and reduced its computational complexity.

E. Fuzzy Clustering Algorithms

Fuzzy clustering is one of the unsupervised learning procedures which are helpful in pattern recognition applications. As the number of various clustering parameters increases its computation becomes more and more hard. Anderson *et al.* investigated GPGPU in order to speed up clustering algorithm as it involves various stages and components that are data independent. In this implementation arrays of input data sets are passed from CPU to GPU as a texture [13]. To calculate the final updated center the whole algorithm is divided into six different subprograms and run on GPU. GPGPU implementation of clustering is found to have better speed performance by a factor of 2 at lower cost. Many heavy other computations can be implemented using the basic idea of this paper.

As discussed earlier, the author used simpler algorithm and offloaded the task of fuzzy clustering to a GPU, however, this approach is not much efficient for large data sets. Therefore, later he incorporated non-Euclidean distance metrics into fuzzy clustering on GPU [14]. Here, NVIDIA 8800 GPU is used along with 32-bit Intel CPU. The results have shown that as the number of samples are increased GPU outperformed CPU with this technique. Computations speedup using this method has improved by almost 2 orders of magnitude. The work can, further, be extended to even larger data sets.

F. Type-2 FLS

1) Interval type-2 FLS for robotic navigation

Type-2 FLSs are comprised of fuzzy sets whose membership values are Type-1 membership functions and called secondary membership functions. Fuzzy computations such as rule implications, aggregation, and defuzzification, etc., become very intensive for ordinary computers [19]. Ngo *et al.* proposed the use of GPGPU for implementation of IT2 FLS to achieve obstacle avoidance behavior of robot navigation [15]. Various stages and components of the algorithm are independent of each other, therefore, possible to be implemented in parallel on GPUs. An FLS consisting of two inputs (the extended fuzzy directional relations and range to obstacle) and one output (angle of deviation) is implemented on NVIDIA Geforce GT540M graphics card having 96 CUDA cores, 1GB of texture memory along with Intel Core i3-2310M2, 1 GHz CPU. Experimental results have shown that with the increase in the number of rules and sample rate the GPU outperforms the CPU. With 8192 sample rate and 512 rules GPU performs approximately 30 times faster than that of CPU. In future, GPGPU based Type-2 FLS implementations can be investigated for better performance to solve engineering problems.

IV. Conclusion

In this paper, we have shown how GPGPU has emerged out as a low cost solution to HPC. Tremendous amount of speedups achieved using GPGPU in implementations of different FLSs is the driving force behind its popularity. In Mamdani FLS, a speed up of approximately a factor of 178 has been obtained on GPU as compared to CPU. FIS implementation runs approximately 51 times faster on GPU than traditional methods used on CPU. Implementation of fuzzy TSK tuning on GPU surpassed the CPU by a factor of around 5 to 6 times. Fuzzy logic based image processing using GPU attained 1.26 times speed improvement. GPU based fuzzy connectedness image segmentation algorithm achieved a speed up factor of 2 to 42 times. The processing time of GPU based algorithm implementation of FAD is 146 times less than corresponding processing time achievable with conventional CPU implementation. Speed of fuzzy clustering on GPU increased over 2 orders of magnitude and on incorporation of non-Euclidean metrics into fuzzy clustering GPGPU has, further, increased the speed up by two orders of magnitude. In fuzzy GPU, gain of 2 to 20 has been obtained. An FLS designed on GPU for robotic navigation with collision avoidance behavior runs 30 times faster on GPU as compared to CPU implementation.

All these FLS implementations using GPGPU (not big in numbers, at this point of time) and their impressive outcomes are sufficient driving force for researcher to investigate this low cost HPC paradigm for more applications.

References

[1] D. Anderson and S. Coupland, "Parallelisation of Fuzzy Inference on a Graphics Processor Unit using the Compute Unified Device Architecture, " *in Proceedings of the UK Workshop on Computational Intelligence (UKCI'08)*, 2008, pp. 1–6.

[2] J. M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial, " *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345–377, 1995.

[3] L. A. Zadeh, "The Concept of a Linguistic Variable and Its Application to Approximate Reasoning". *Information Sciences*, vol. 8, no. 3, 1975, pp.199-249.

[4] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications.* NY: Academic Press, 1980.

[5] O. Castillo and P. Melin*, 3 Type-2 Fuzzy Logic.* Springer, 2008.

[6] N. N. Karnik and J. M. Mendel, "Operations on Type-2 Fuzzy Sets, " *International Journal on Fuzzy Sets & Systems*, vol. 122, pp. 327–348, 2001.

[7] N. Harvey, R. Luke, J. M. Keller, and D. Anderson, "Speedup of Fuzzy Logic Through Stream Processing on Graphics Processing Units, " in *IEEE Congress on Evolutionary Computation, 2008,* pp.3809–3815.

[8] B. B. Ferreira and A. J. Cruz, "A Parallel Method for Tuning Fuzzy TSK Systems with CUDA, " *SBC–Proceedings of SB Games, Brazilian Computer Society (SBC)*, pp. 5–8, 2012

[9] D. Defour and M. Marin, "Fuzzy GPU: A Fuzzy Arithmetic Library for GPU, " in *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on.* IEEE, 2014, pp.624–631.

[10] R. H. Luke III, D. Anderson, J. M. Keller, and S. Coupland, "Fuzzy Logic based Image Processing using Graphics Processor Units", in *IFSA/EUSFLAT Conference*, 2009, pp.288–293

[11] Y. Zhuge, J. K. Udupa, K. C. Ciesielski, A. X. Falc˜ao, P. A. Miranda, and R. W. Miller, "GPU-based Iterative Relative Fuzzy Connectedness Image Segmentation, " in *SPIE Medical Imaging*. International Society for Optics and Photonics, 2012, pp. 831 604–831 604.

[12] R. d. J. D. Coello, F. d. J. S. Lugo, A. C. Atoche, and J. O. Aguilar, "GPU Implementation of Fuzzy Anisotropic Diffusion." in International Conference on Information and Communication Technologies and Applications (ICTA), 2012

[13] D. T. Anderson, R. H. Luke, and J. M. Keller, "Speedup of Fuzzy Clustering Through Stream Processing on Graphics Processing Units, " *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 4, pp. 1101–1106, 2008.

[14] D. T. Anderson, R. H. Luke, and J. M. Keller, "Incorporation of Non-euclidean Distance Metrics into Fuzzy Clustering on Graphics Processing Units, " in *Analysis and Design of Intelligent Systems using Soft Computing Techniques*. Springer, 2007, pp. 128–139.

[15] L. T. Ngo, D. D. Nguyen, C. M. Luong *et al.*, "Speedup of Interval Type-2 Fuzzy Logic Systems based on GPU for Robot Navigation, " *Advances in Fuzzy Systems*, vol. 2012, p. 4, 2012, pp. 1-11

[16] M. Khosla, R. K. Sarin, M. Uddin, and S. Singh, A. Khosla, "Realizing Interval Type-2 Fuzzy Systems with Type-1 Fuzzy Systems", in *Cross-Disciplinary Applications of Artificial Intelligence and Pattern Recognition: Advancing Technologies*, IGI Global, Hershey, Pennsylvania, USA, 2012, pp. 412--427.

[17] J. M. Mendel, R. I. John and F. Liu, "Interval Type-2 Fuzzy Logic Systems Made Simple", *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 6, 2006, pp.808–821.

[18] S. Singh, S. Singh, V. K. Banga, D. Chauhan, "CUDA for GPGPU Applications - A Survey", in Proc. National Conference on Contemporary Techniques & Technologies in Electronics Engineering, Murthal, Sonepat, India, March, 2013, pp.189--192.

[19] S. Singh, J. S. Saini, V. Mutneja, N. Gill, "Mobile Robot Navigation using IT-2 FLS", in Proc. IEEE National Conference on Applications of Intelligent Systems (AIS-2008), Sonepat, India, March, 2008.