Design and Implementation of Rough Set Algorithms on FPGA: A Survey

Kanchan Shailendra Tiwari
Research Scholar, ECE Dept. VNIT, Nagpur
Asst. Professor, E&TC Dept. MESCOE,
Pune, India

Ashwin. G. Kothari
Associate Professor, ECE Dept.
VNIT
Nagpur, India

***Abstract*—Rough set theory, developed by Z. Pawlak, is a powerful soft computing tool for extracting meaningful patterns from vague, imprecise, inconsistent and large chunk of data. It classifies the given knowledge base approximately into suitable decision classes by removing irrelevant and redundant data using attribute reduction algorithm. Conventional Rough set information processing like discovering data dependencies, data reduction, and approximate set classification involves the use of software running on general purpose processor. Since last decade, researchers have started exploring the feasibility of these algorithms on FPGA. The algorithms implemented on a conventional processor using any standard software routine offers high flexibility but the performance deteriorates while handling larger real time databases. With the tremendous growth in FPGA, a new area of research has boomed up. FPGA offers a promising solution in terms of speed, power and cost and researchers have proved the benefits of mapping rough set algorithms on FGPA. In this paper, a survey on hardware implementation of rough set algorithms by various researchers is elaborated.**

Keywords—Rough set theory; Discernibility matrix; reduct; Core; FPGA; classification

I. INTRODUCTION

Rough set theory(RST), by Zdzisław Pawlak, is a powerful mathematical tool, for discovering data dependencies by reducing the number of attributes contained in a data set using the data alone, without requiring any further additional information like degree of membership, probability etc. as required in fuzzy or in probability theory[1]. It is not an alternative to classical set theory but rather embedded in it. It provides efficient algorithms for finding hidden patterns in data, minimal sets of data (data reduction), evaluating significance of data, and generating sets of decision rules from data. The rough set approach is easy to understand, offers straightforward interpretation of obtained results, most of its algorithms are particularly suited for parallel processing. It is considered as one of the first non-statistical approach in data analysis [2]. Its methodology is concerned with the classification and analysis of imprecise, uncertain, vague or incomplete information and knowledge. The conceptual foundation of rough set data analysis is the consideration that all perception is subject to granularity and the ability to classify is at the root of human intelligence [3].

RST has been widely used in machine learning, data mining, and artificial intelligence successfully. Various software tools like ROSE, RSES, and ROSETTA [4-6] etc. are used for generating reduct, cores, and meaningful rules. These purely software program offer users a relatively high level of versatility and can handle any type of algorithm but the biggest and important issue is deterioration in the performance as the size of datasets increases. The software execution time becomes relatively slow while handling large real time datasets since the processor is not specially optimized for it. With the advent of digital technologies, Internet of Things, social media, etc. online storage of data has increased exponentially. It's need of the hour to process data in real time and at a faster rate. Recently, there has been a growing interest amongst researchers in developing a dedicated hardware for RST using FPGAs. The advantage of using a dedicated hardware is huge acceleration in terms of speed as they relieve main processor from the computational overheads. There are several such accelerators already available commercially in markets like Graphics Processing Units (GPUs), Digital Signal Processor (DSP), Fuzzy Processor. A dedicated hardware of rough set modules tends to be much faster than their software counterpart. The growth of VLSI industries had led to significant improvement in FPGAs in terms of resources available, speed, cost, and re-programmability etc. motivating researchers to choose it as one of the most viable solution.

In this paper in section 2, the basics of rough set theory are presented. In Section 3, need for hardware accelerator is discussed while section 4 covers the current status of art in the design of Rough Set Processor (RSP) by various authors followed by conclusion in section 5.

II. ROUGH SET PRELIMINARIES

The information in the world surrounding us is often imprecise, incomplete and uncertain. The human's ability of thinking and concluding widely depends on this information. In order to draw conclusion, one has to process this incomplete and imprecise data [7].

A soft computing tool mimics human decision making system and hence gives more promising results while handling such data. The various soft computing tools are fuzzy theory, neural network, genetic algorithms, rough set theory, etc. Rough set and fuzzy set theory are complementary to each other. RST is an effective tool for mining deterministic rules from a database. The rough set philosophy is founded on the assumption that with every object of the universe of discourse we associate some information i.e., knowledge is associated, through which classification can be achieved. It is based on

the idea that lowering the degree of precision in the data makes data pattern more perceptible [7]. The main motto of Rough Set theory is "Let the Data Speak for themselves". RST gives more formal framework for discovering facts from imperfect data. It gives results in the form of classification or decision rules derived from a set of examples.

Objects characterized by the same information are indiscernible (similar) in view of the available information about them. The indiscernibility relation generated in this way is the mathematical basis of rough set theory. Any set of all indiscernible (similar) objects is called an elementary set (neighborhood), and forms a basic granule (atom) of knowledge about the universe (fig.1). Any union of elementary sets is referred to as crisp (precise) set - otherwise the set is rough (imprecise, vague). Some of the Rough set related terms are presented below [7][8]:

A. Information System

The basic vehicle for data representation in the rough set framework is an information system (IS). An IS is a table, listing attributes of objects. Each row represents objects while each column specifies its attributes or features. Formally IS can be defined as IS = (U, A) where U is finite set of objects, $U=\{x_1, x_2, x_3, \ldots, x_n\}$; and A is a finite set of attributes (features, variables), the attributes in A are further classified as condition attributes C and decision attribute D, such that $A=C \cup D$ and $C \cap D=\emptyset$ (empty). Table 1 shows an example of a typical information system.

TABLE I. AN INFORMATION SYSTEM

Objects.	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	d
x_1	1	1	0	0	1	1	0	0	1
x_2	0	0	1	1	1	1	1	0	2
x_3	1	0	1	1	1	1	0	0	3
x_4	1	0	0	0	1	1	1	1	4
x_5	1	1	1	1	0	0	0	1	2
x_6	1	0	1	0	0	1	1	1	3
x_7	1	1	1	0	0	0	1	1	4
x_8	0	0	0	1	0	1	0	0	1

B. Decision Attributes

These are those attributes, which absolutely decide to which class the object belongs. In an IS shown in Table 1, d column is decision attribute column. The value of d, in it ranges from 1 through 4. Hence above IS is a 4 class system.

C. Condition Attributes

These are those attributes which do not absolutely decide the class to which the object belongs, but helps to decide. IS with distinguished decision and condition attributes are called decision tables. In Table 1, c_1, c_2, c_3---c_8 are condition attributes of 8 objects.

D. Upper Approximation ($\bar{A}(x)$)

Upper Approximation is a description of the objects that possibly belong to the subset of interest.

E. Lower Approximation ($\underline{A}(x)$)

It consists of those objects that can be with certainty classified as belonging to X. It is also known as POS(X).

F. Boundary Region

A set is said to be rough if its boundary region is non-empty, otherwise the set is crisp. It is also known as BR(X) Whereas U - $\bar{A}(x)$ is known as NEG(X). If the boundary region is a set $X = \emptyset$ (empty), then the set is considered "Crisp", otherwise, if the boundary region is a set $X \neq \emptyset$ the set X "rough" is considered.

G. Indiscernibility relation

Indiscernibility relation is a central concept in RST and is considered as a relation between two objects or more, where all the values are identical in relation to a subset of considered attributes. Indiscernibility relation is an equivalence relation, where all identical objects of set are considered as elementary.

H. Discernibility Matrix

An information system can also be presented in terms of a discernibility matrix. A discernibility matrix is a square matrix in which rows and columns are objects, and cells are attribute sets that discern objects. Two objects are considered discernible if and only if they have different values for at least one attribute. The discernibility matrix, denoted by M, for a decision table DT, of an IS is given as –

$$c_{ij} = \begin{cases} \Phi; & f_D(x_i) = f_D(x_j) \\ a \in A; & a(x_i) \neq a(x_j), f_D(x_i) \neq f_D(x_j) \end{cases} \quad (1)$$

A discernibility function can be constructed from discernibility matrix by OR-ing all attributes in c_{ij} and then AND-ing all of them together. After simplifying the discernibility function using absorption law, the set of all prime implicants determines the set of all reducts of the information system. However, simplifying discernibility function for reducts is a NP-hard problem. In Table 2 partial discernibility matrix for IS shown in Table 1 is tabulated.

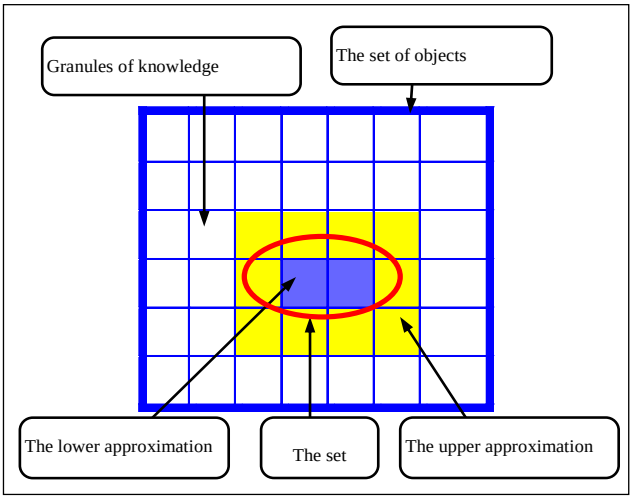

Fig. 1. Rough Set Concept Illusttration.

TABLE II. PARTIAL BINARY DISCERNIBILITY MATRIX

Objects.	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
x_{12}	1	1	1	1	0	0	1	0
x_{13}	0	1	1	1	0	0	0	0
x_{14}	0	1	0	0	0	0	1	1
x_{15}	0	0	1	1	1	1	0	1
x_{16}	0	1	1	0	1	0	1	1
x_{17}	0	0	1	0	1	1	1	1

I. Reduct and Core

The reduct and the core are important concepts in rough sets theory. A reduct is any minimal subset of condition features, which discerns all pairs with different decision values and is complete if the deletion of any attribute of a reduct will make at least one pair of objects with different decision attribute values indiscernible. The intersection of all reducts is called the core of the decision table. Discernibility matrix and Positive region based methods are more popular for computation of reducts in RST. Reducts can be of dynamic types too. Dynamic reducts are just a subset of all reducts which are derivable both from the original decision table and from the majority of randomly chosen decision sub-tables. Dynamic reducts gives dynamic rules.

J. Inconsistent Decision Table:

A decision table is inconsistent if for a given pair of object, all condition attributes are same but differ in decision attribute i.e. belong to two different classes. A medical database of 6 patients having symptoms of flu is shown in Table 3. The symptoms of flu are conditions attributes, which includes headache, muscle-pain, and temperature etc. while whether the patient is suffering from flu or not (1 or 0) is indicated by last column, also called as decision attribute. In Table 3, object 2 and 5 makes database inconsistent.

TABLE III. INCONSISTENT DECISION TABLE

Patients	Attributes			Decision
	Headache	***Muscle-pain***	***Temperature***	***Flu***
P_1	No	Yes	High	No
P_2	Yes	No	High	Yes
P_3	Yes	Yes	Very High	Yes
P_4	No	Yes	Normal	No
P_5	Yes	No	High	No
P_6	No	Yes	Very High	Yes

III. NEED OF HARDWARE ACCELERATORS

In data mining, processing of large volumes of data using complex algorithms is increasingly common. There are numerous applications like image processing, speech processing, artificial intelligence, analyzing experimental data etc. which demands fast processing of high volumes of data.

Computers are able to handle a wide variety of applications. Since the design and development of computers from 1940, there has been exponential growth in its performance for decades. This growth has been further complemented by a combination of improvements in implementation technology, architectural innovations, and compiler optimizations. However, as computers becomes even faster, new applications empowered by technology arise, which demands development of new technologies [9]. In addition to those continuous improvements, designers have relied on solutions based on special architectures to accelerate the performance of these applications, with processing units exploiting their common features such as parallelism, repetitive tasks or intensive mathematical processing. Traditionally, these solutions have been of two types:

A. Parallel Processing Computers With Parallel Processors

During the last few decades, traditional general-purpose single-core CPUs has shown a remarkable growth due to the multiple improvements in VLSI technologies. This growth was marked by the reduction in size of transistors, increase in the frequency of processor as per Moore's law and hence software performance also improved continuously for decades. However, the gain in the performance of conventional single core CPU has diminished as the VLSI system performance hit the memory wall, power wall [10] and instruction-level parallelism (ILP) wall. The memory wall refers to the increasing gap between processor and memory speeds. This demanded increase in size of cache for hiding memory access latencies [11]; thus making memory bandwidth a bottleneck in performance. The power wall refers to power supply limitations and thermal dissipation limitation. For the silicon lithography below 90nm, the static power from leakage current surpass dynamic power from circuit switching. Power density has become the dominant constraint in chip design, and limits the clock frequency growth [12]. The performance, cost, and reliability of modern computer systems and data centers are dictated by the management of their limited energy and thermal budgets [13]. The ILP wall refers to the rising difficulty in finding enough parallelism in the existing instructions stream of a single process. Increasing cache size or introducing more ILP yields too little performance gain compared to the development cost [14]. Together, these three walls reduce the performance gains expected for single-core general-purpose processors.

With current technology, even though the number of transistors is increasing, but the clock speeds are flattening as shown in fig.2. In order to overcome the problems posed by power and ILP wall, the computer industry shifted from single core processors to multiple parallel processing units. This showed the beginning of a paradigm shift towards parallel hardware architectures. CPU manufacturers used the improved processes to fit more and more CPU cores onto each device, producing generations of many-core processors, each running at about the same clock frequency as their predecessors.

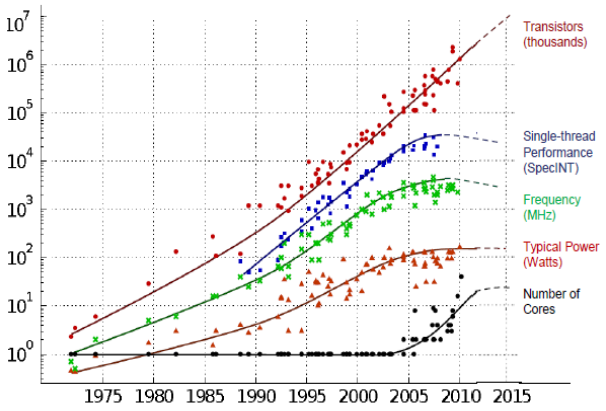

Fig. 2. Moore's Law- The number of transistors and power consumption is constantly increasing, while the frequency is flattening. (Source- Taken from Kunle Olukotun and Herb Sutter)

However, conventional computer programs are described as sequentially executed instructions and cannot easily be adapted for a multi-processor environment. This condition prevents a potential speed-up due to the problem of finding enough parallelism in the software. The speedup S from using N parallel processing units is calculated using Amdahl's law [15] as

$$S = \frac{1}{(1-p)+\frac{p}{N}} \qquad (2)$$

where p is the fraction of the sequential program that can be parallelized.

If one assumes that 50% of the sequential code can be executed on parallel processors, then speedup will still be limited to a modest factor of 2, no matter how many parallel processors are used. Parallel architectures have a promising future, but will require new design approaches and programming methodologies to enable high system utilization. This means that for faster execution, one must actively seek alternative ways to speed up the software.

B. Accelerators

In order to exploit the parallelism and distribute the computation amongst several processing cores, software execution style should change from sequential to parallel. This opens up the playfield for new types of processing resources to complement the traditional CPU architecture. Recently, market is dominated by cost-efficient accelerators available from several vendors as common off the-shelf (COTS) products [16]. Accelerators are specialized processors that can be used to speed up specific processing tasks and they complement conventional architectures. Accelerators with CPUs, forms a hybrid computing system or Multi-Processor Systems-on-Chip (MPSoCs), where each processing resource executes the parts of the software for which it delivers the best performance. Currently, heterogeneous MPSoCs are becoming the de-facto standard for embedded system design. Such system usually is composed of several general purpose processors, digital signal processor and hardware accelerators interconnected through various communication mechanisms for accelerating specific part of an application. This results in greatly increased system performance.

The main competitors for the COTS accelerator market are Field Programmable Gate Arrays (FPGAs) and Graphics Processors (GPUs). These devices have strong mass markets in the high performance computing fields. Acceleration continues to be a great necessity in this new scenario dominated by multicore processors and clusters built with them, because of the following reasons:

- Optimum performance for all types of applications is not given by General Purpose Processors, even if multicore technology is used.
- There are certain applications like single thread applications, embedded systems, etc., where significant acceleration is not achieved by using conventional multicore technology.
- The complexity and huge size of digital circuit causes the run time of software to become unreasonably large as these problems are NP-hard.
- To reduce execution time.
- To offload the general purpose CPU.
- To offer special features for easy use.

Therefore, while the use of specific parallel processors computing has declined, new solutions continue to appear in the field of hardware accelerators. Accelerators can be realized using different technologies like DSP, GPGPU, ASIC, FPGA.

They all differ in architectures and are suited for different applications.

1) Digital Signal Processor (DSP)

- A DSP is a processor system optimized to implement signal processing at very high speed.
- DSP's include a specialized architecture which allows parallel processing at the instruction level; this is called SIMD (Single Instruction Multiple Data).
- There are fixed and floating point DSP's available in market.
- Parallel instructions are used with special assembler instructions included in C program.
- The DSP Blackfin 609 is a fixed point DSP based in a Dual-Core processor working up to 1GHz. The Blackfin arithmetic unit allows the execution of multiple operations in parallel: up to four 8-bit video ALUs or two multiplications and 2 accumulations of 32/40-bits.

2) Application Specific Integrated Circuit (ASICs).

- ASIC is basically a circuit designed for a specific use rather than a circuit designed for general purposes
- ASIC designs offer a very attractive solution for many high volume applications.

- The design using ASIC offers better performance, density and power consumption when compared to an FPGA.
- ASIC prototyping can be done using FPGAs, which allows taking advantage of FPGAs re-programmability.
- However, the cost of prototyping is quite high increasing the Nonrecurring Engineering (NRE) costs depending upon the design, complexity and method of implementation.
- Also, they do not offer any flexibility, as the task they perform cannot be modified.
- Hence, their use for acceleration purpose is quite limited.

3) General Purpose Graphical Processing Units (GPGPUs).

- Graphics Processors are highly parallel processors capable of running thousands of threads simultaneously. Threading is handled automatically by the hardware thread manager. The programmer does not have direct control of the processors of the GPU; everything is done through Application Programming Interfaces (API).
- They are special types of processor dedicated for graphics operation in game consoles and computers.
- They are an order of magnitude faster on floating point operations.
- Recently GPGPUs have been specifically developed with the computational precision required for finite element analysis solutions as well as the computational power to effectively complement the performance of the latest CPUs.
- With hundreds of low-power cores on a single socket, they have the potential to dramatically increase computing capacity, provided that the compute workload will fit in the available memory of the GPGPUs.
- However, the applications with complex feedback loop, and control or extensive bit handling is not suitable for GPGPUs implementation. The high power consumption of GPGPUs restricts their usage to certain applications.
- GPGPUs are difficult to program for general-purpose uses.
- In the current market there are three principal GPU providers: NVidia, Intel, and AMD.

4) Field Programmable Gate Arrays (FPGAs)

- FPGA is semiconductor device, invented by Xilinx co-founder, Ross Freeman, in 1984.
- FPGAs generally consist of sets of flexible gates, registers, and memories whose function and interconnection are controlled through the loading of SRAMs (Static Random Access Memory).
- FPGA can be programmed either statically (between applications) or dynamically (during an application) without the addition of physical hardware elements.
- It is intended to fill the gap between the hardware (ASICs) and software (General Purpose (GP) Processors), achieving potentially much higher performance than software, while maintaining a higher level of flexibility than hardware
- The main resources available in the current FPGAs are hard Processors, RAM memory, Slices, DSP Slices, Multipliers, Gigabit transceivers, Triple-Speed Ethernet MAC, PCI express, Phase Locked Loop (PLL), etc.
- FPGAs tend to operate at relatively modest clock rates measured in a few hundreds of MHz, but they can perform sometimes tens of thousands of calculations per clock cycle while operating in the low "tens of watts" range of power.
- Improvements in FPGAs have driven a huge increase in their use in space, weight and power (SWaP) constrained embedded computing systems for military and aerospace applications. They are ideal for addressing many classes of military applications, such as Radar, SIGINT, image processing and signal processing where high-performance DSP and other vector or matrix processing is required.
- FPGAs seem to gives an unbeatable edge over a microprocessor as they can provide 50 to 100 times the performance per watt of power consumed than a microprocessor.
- FPGA offers field reprogrammability. A new bit stream file can be uploaded remotely.

The advantages and disadvantages of FPGA with respect to other available technologies are presented in table 4. In case of the ASIC, the fabrication cost is reduced if chip is produced in mass; however for unit production ASIC design is expensive. FPGA combines many benefits of both software and ASIC implementations. Like software, the mapped circuit is flexible, and can be reconfigured over the lifetime of the system. FPGAs therefore have the potential to achieve far greater performance than software as a result of bypassing the fetch-decode-execute operations of traditional processors, and possibly exploiting a greater level of parallelism. Creating parallel programs implemented in FPGAs is not trivial. Fig. 3 [17] summarizes the application fitness categorization. Hence it is concluded that FPGA is best choice for implementation of rough set algorithms as it out performs with respect to other available technologies.

	GPGPU	FPGA
Good to Fit ↕ Bad Fit	No inter-dependences in the data flow and the computation can be done in parallel	Computation involves a lot of detailed low level hardware control operation which cannot be efficiently implemented in high level languages, such as bit operation
	Applications contain a lot of parallelism, but involve computations which cannot be efficiently implemented on GPU	A certain degree of complexity is required, and the implementation can take advantage of data streaming and pipelining
	Applications have a lot of memory access and have limited parallelism	Applications that require a lot of complexity in the logic and data flow design

Fig. 3. Application Characteristic Fitness Categorization

TABLE IV. TECHNOLOGY COMPARISON

Features	Platforms				
	DSP	*GPU*	*ASIC*	*FPGA*	*GPP*
Size	Medium	-	High	High	Medium
Power	Medium	-	High	Low	Medium
Flexibility	High	High	-	High	High
Reliability	Low	Low	High	Medium	Low
Parallelism	Low	Medium	High	High	Low
Operation Frequency	Medium	High	High	Low-Medium	High
Design complexity	Medium	Low	-	-	High
Cost	Medium	Low	High	Low	High

IV. CURRENT STATE OF ART

A. Z. Pawlak Concept of Rough Set Processor

The concept of Rough Set Processor (RSP) is put forth by Z. Pawlak (RSP) in [18]. He stated that RSP can be used as an additional fast classification unit in ordinary computers or as an autonomous learning machine. In latter case, RSP can replace neural networks. He stated that each row of a decision table induces a rule, which specifies the actions if some conditions are satisfied. If a decision rule uniquely determines a decision in terms of condition attributes then that rule is certain otherwise it is uncertain. According to him, decision rules are closely associated with concept of approximation in rough set theory. Lower approximation are described by *certain* decision rules while upper approximation by *uncertain* decision rules. He associated the two conditional probabilities called, *uncertainty* and *coverage coefficient* with each decision rule. The *certainty coefficient* expresses the probability that an object belongs to the decision class specified by the decision rule, if it satisfies the condition of the rules. The *coverage coefficient* gives the conditional probability of the reasons for a given decision. He proved that the *certainty* and *coverage* coefficient satisfy Bayes' theorem and it can be used for drawing conclusion from data. This idea is used as a foundation for RSP. The computation of certainty and coverage factors of decision rules is dependent on strength of decision rules. The strength can be computed from data or can be a subjective assessment. The concept of flow graph i.e. a directed acyclic graph is associated with decision table. In that graph, to every decision rule, a directed branch connecting input node with output node is assigned. The strength of the decision rule represents a throughflow of the corresponding branch. The classification of objects is done by finding the maximal output flow in the flow graph whereas, the explanation of the decisions is connected to the maximal input flow associated with the given decision. He proposed requirement of a special microprocessor for doing all above mentioned computation. According to him, RSP should perform operations pointed out by the flow graph of a decision table i.e. first computation of strengths from the support of decision rules, and then certainty and coverage factors of all rules should be computed. All these parameters are stored and computed subsequently in a format of word structure as shown in Fig. 4. Decision table will store condition and decision attributes of objects, Decision rule register will compute meaningful rules from data while arithmetic block will perform arithmetic operation of computing strength, coverage and certainty factors as shown in Fig. 5. This idea, however, is not realized on programmable logic devices.

Condition attribute	Decision attribute	Support	Strength	Certainty	Coverage

Fig. 4. Word structure.

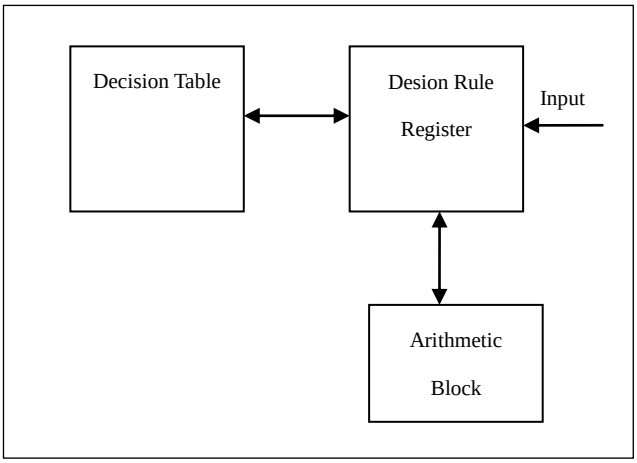

Fig. 5. RSP Structure.

B. Towards PRSComp

Authors in [19] put forward the concept of describing rough set methods using cellular networks. They designed a

device for parallel processing of rough set algorithms and called it as Parallel Rough Set Computer (PRSComp). Cellular network is a matrix of interconnected elements of the same type, wherein each cell is treated as a single processor and a set of control registers. The cellular network based on rough sets transforms the input data set to the matrix and performs basics operation of rough sets using matrix notations. In [19] authors have used all basic notions of rough set (indiscernibility relation, upper and lower approximation, reducts and core calculations) for implementation of PRSComp. Authors have given pseudo code for all basic routines also.

C. Lewis T. Idea as Learning Machine

Authors in [20] built a Universal Logic Machine (ULM) based on the principles of constructive induction and RST. It is a self-learning rough set model based on the concept of cellular networks by [19]. It is thought of an early prototype of data mining machine which will not only be able to collect data from online databases, but also from industries, military and other real time applications. The authors presented a preliminary work on design and implementation of a single instruction multiple data (SIMD) computer to implement RST operations. RST can be effectively used in logic minimization and data mining. They identified that some subsets of RST are isomorphic with some subsets of logic synthesis and decomposition theories; hence their mutual relationship can be investigated, leading to synergies of concepts. For example the powerful logic concepts of rough set theory can be linked with efficient algorithms and data structures developed in logic synthesis for EDA. According to them the RST algorithms have a natural high parallelism and high possible speed-ups. Using a fast prototyping tool, the DEC-PERLE-1 board based on an array of Xilinx FPGAs, a virtual SIMD processor that accelerates the learning (design) of optimized multi-valued logic nets using the concept of cellular networks has been developed. They have proposed the principles of learning hardware that will use previous human problem-solving experience and apply mathematical algorithms, problem-solving strategies rather than relying only on neural network and genetic algorithm.

A solution to a given problem is achieved by partitioning it in two phases: the phase of learning and the phase of using the knowledge. The hardware processor (parallel rough set computer) is responsible for creation of logic network description using logic or mathematical algorithms. The optimally constructed network is mapped on FPGA using EDA tool. The knowledge of machine is stored in memory. While solving the new problem under the supervision of software program in the main processor, the hardware switches between various learned nets, depending on rules. Since network has to solve new problems, hence new datasets and training decisions are accumulated and the network is repetitively automatically redesigned. The old network can serve as a platform for redesigning of new network or new network can start from scratch to avoid any bias.

Lewis implemented basic rough set operation of basic category, upper approximation, and lower approximation, indispensable and external comparison. Authors demonstrated the working of all above mentioned algorithms on a learning machine called as parallel rough set computer (PRSComp) and its architecture is shown above in Fig. 6. In Fig. 6 E is word selection register, C is comparand register and CM is column mask register. He proposed a machine consisting of m by n primitive processor, in which each of the processors is connected to its neighboring four processors as well as to global control signals. Each processor performs the same operation defined by the instruction at that time. PRSComp operates as SIMD (single input multiple data). The input data is mapped on these processors as a binary matrix of size m*n wherein each processor operates on one bit of it at a time. They utilized various registers for doing all these operations. In this paper, there is no discussion on time complexity, space complexity. The author however has put forth the problems posed by purely genetic and artificial neural network and justified that rough set theory is an appropriate solution for handling those problems.

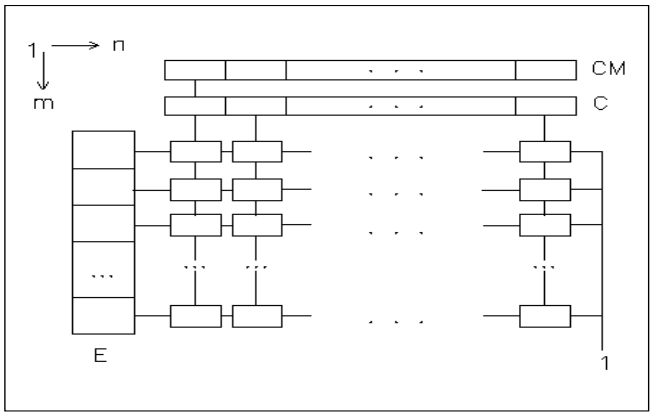

Fig. 6. PRScomputer architecture.

D. Kanasugi Discernibility Matrix approach

Authors in [21] presented a design of architecture of rough set processor in 2001 (shown in fig.7). It is used for solving large-scale problem in real time. The main blocks in their architecture are discernibility matrix maker, core selector, covering unit, reconstruction unit, registers, cache memory, controller and bus interface. The execution process is divided into two parts: pre-process and main process. In pre-process, some sparse terms are selected as cores and then implying relation reduces the input logic functions. In the main process, input logic function is converted into the sum of products form.

The block of discernibility matrix maker is not dealt in this proposed work. The core selector unit selects data whose sum is minimum and transfers its row number to core number register unit. Core unit reduces data using implying function. Reconstruction unit searches for dominant variables from input logic function and then reconstructs the important rules from it. Memory interface is identified as a potential bottleneck in the design. The work of [22] is an extension of [21]. In [21], only design has been proposed whereas in [22], synthesis, simulation and implementation on SPARTAN 3E board of Xilinx is presented. They minimized the discernibility matrix by obtaining a reduced discernibility function. The outputs of system are small logical functions

representing important decision rules. Authors have developed a co- processor, which will be interacting with memory for data retrieval and storage purpose. The system depends on external data source for creation of large logical functions from data base for correct operation and algorithm is based on approximation technique. Their co-processor is capable of dealing with objects of size 1000,000 and 2032 attributes. They have dealt with binary attributes, leaving discretization process, a task for future development. They have shown that their proposed processor is ten times faster than PC, though the clock frequency is about 70 times slower. There is no discussion on time complexity and space complexity. However, their algorithm is based on computing discernibility matrix and discernibility function, whose time complexity will no longer be less than O ($|U|^2|A|^2$).

E. G. Sun's FPGA implementation of RST

G. Sun in his paper [23] has implemented Rough set theory algorithms on FPGA in 2011. Author has provided a new and effective method for hardware fault diagnosis and verified the effectiveness of method through simulation. He has made use of genetic algorithm along with rough set theory and presented a case study of nonlinear aircraft model. He implemented discretization block, based on dependency degree. The breakpoints are deleted based on dependency degree. The reducts are calculated using genetic algorithm. The simulation results using Modelsim for discretization and attribute reduction has been presented. The algorithms are not purely based on RST; rather it is hybridization of rough set with genetic algorithms.

F. Maciej Kopczynski ´s et al. computation of reduct and core on FPGA

Maciej Kopczynski et al. in their paper [24 - 26] presented reduct and core generation algorithm based on discernibility matrix. They have presented hardware solution architecture for binary decision table. They have discussed architecture of discernibility and reduct block. They used VHDL simulator and the development board equipped with an Altera FPGA during the research. The reduct generation algorithm is simple and based on attribute count frequency [25]. The algorithm gives super reduct, however it does not discusses the case of breaking tie between two attributes having the same count value. They have also compared the time required for execution of reduct and core generation on software and hardware for varying size of database. They have randomly generated the binary database. Their results show a significant increase in the speed of data processing. In [26], they have shown three variants of discernibility matrix implementation. Authors have shown time required for computing reducts and cores for all three methods. The issue of dealing with larger databases is not handled.

G. K.S.Tiwari's et.al. Hardware Implementation

Tiwari et.al in their work [27] presented architecture for computing reduct using binary discernibility matrix. They have used Xilinx software and Spartan 3 FPGA. They have proposed a Rough Set Machine which generates rules for classification applications. The classification task concentrates on predicting the value of the decision class for an object among a predefined set of classes' values. This rough set machine uses the concept of discernibility matrix for calculating the reduct, and using these reduct it generates the rules which are used for classifying the objects. The Reduct block is synthesized and downloaded on FPGA in [28]. The architecture of binary discernibility matrix is shown in fig.8. In [29]; Quick reduct algorithm is used for computation of reduct for a medical database.

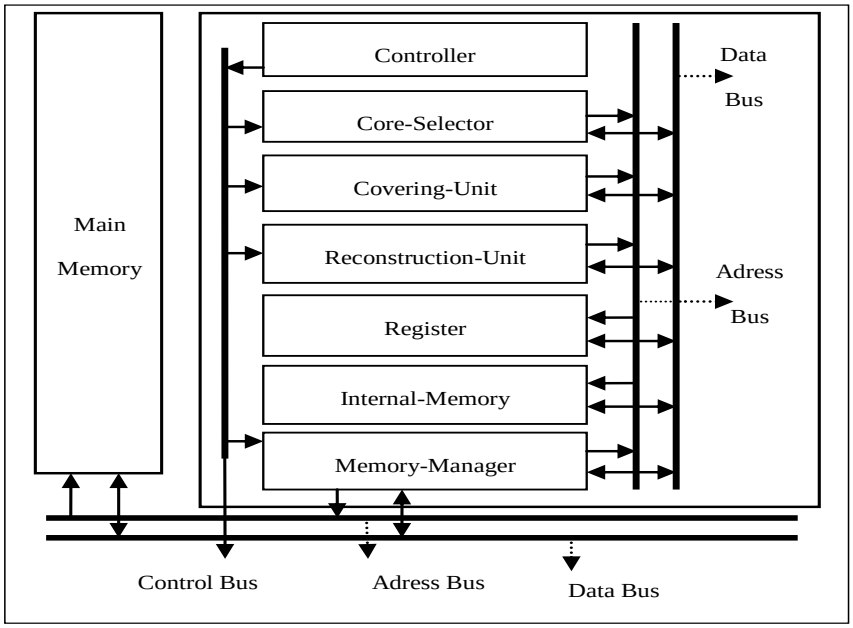

Fig. 7. Kanasugi's Proposed Block Diagram of Rough Set Processor.

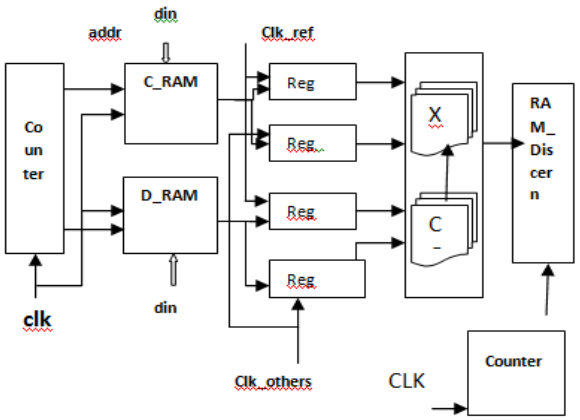

Fig. 8. Binary discernibility Matrix

TABLE V. SUMMARY

Sr.No.	Authors	Year	Brief Summary
1	Mieczyslaw Muraszkiewicz, and Henryk Rybinski	1994	Concept is based on Indiscernibility relation, Lower and upper approximation
2	Lewis et.al	1999	Self-learning hardware model based on cellular concept, Implementation done Xilinx board.
3	Kanasugi	2001	Algorithm is based on Discernibility matrix
4	Z.Pawlak	2004	Decision Flow graph used for representing tables.
5	Kanasugi and Mitsuhiro Matsumoto	2007	Discernibility matrix based algorithm proposed and implemented on Spartan 3E Board.
6	G Sun et. al	2011	Genetic based attribute reduction system; discretization is based on dependency approach of RST.
7	K.S.Tiwari et.al	2011	Concept of discernibility matrix used for generation of reducts and rules.
		2012	Pipelining and use of Dual port RAM as a part of extension.
		2013	Quick Reduct algorithm based on dependency function is implemented and simulated using ISIM.
8	Maciej Kopczynski et. al	2013	Computation of short reduct and core based on discernibility matrix. Huge acceleration achieved.
		2014	Discernibility matrix built using three different methods

V. CONCLUSION

In this paper a survey on hardware implementations of Rough set algorithm is presented. It is summarized in brief in table 5. A lot of research work is carried out on rough set theory using software; however hardware implementation is still not much explored. With exponential growth in quantity of data collected, its need of hour to process data fast, and extract meaningful rules from it. FPGA offers a promising solution to deal with such kind of problems as rough set algorithms are inherently parallel. Thus these algorithms can be effectively mapped on FPGA.

REFERENCES

[1] Zdzisław Pawlak, Andrzej Skowron,"Rudiments of rough sets," Information Sciences, vol. 177, January 2007, pp. 3-27.

[2] Zdzisław Pawlak , "Rough Sets," International Journal of Computer and Information Sciences, vol.11, September 1982, pp. 341–356.

[3] Zdzisław Pawlak, Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, Boston, London, 1991.

[4] Bart lomiej Predki et al, " ROSE - Software Implementation of the Rough Set Theory ," Rough Sets and Current Trends in Computing, L. Polkowski and A. Skowron ,Eds., LNCS 1424, Springer-Verlag Berlin Heidelberg 1998, pp. 605-608

[5] Andrzej Skowron et al. Logic Group, Institute of Mathematics, Warsaw University, Poland,1994, Accessed Aug. 2010. http://logic.mimuw.edu.pl/~rses/.

[6] M Kierczak et al., ROSETTA Development Team, 2009, Accessed Aug. 2010. http://www.lcb.uu.se/tools/rosetta/.

[7] B. Walczak, D.L. Massart, "Rough sets theory", Chemometrics and Intelligent Laboratory Systems, vol. 47/1, April,1999, pp. 1–16.

[8] Silvia Rissino, Germano Lambert-Torres "Rough Set Theory – Fundamental Concepts,Principals, Data Extraction, and Applications," *Data Mining and Knowledge Discovery in Real Life Applications*, Julio Ponce , Adem Karahoca,Eds, I-Tech Education and Publishing,2009

[9] Altera. Accelerating high-performance computing with fpgas. white paper, Altera, October 2007.

[10] K. Gulati and S. P. Khatri. Hardware Acceleration of EDA Algorithms: Custom ICs, FPGAs and GPUs. Springer, 2010

[11] D. A. Patterson. Latency lags bandwith. Commun. ACM, 47(10):71–75, 2004

[12] J. Shalf. The new landscape of parallel computer architecture. journal of Physics, 78, 2007.

[13] J. Carter and K. Rajamani. Designing energy-efficient servers and data centers. IEEE Computer, 43(7):76–78, 2010

[14] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli. State-of-the-art in heterogeneous computing. Scientific Programming, pages 1–33, 2010.

[15] Amdahl, Gene M. "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, Reprinted from the AFIPS Conference Proceedings, Vol. 30 (Atlantic City, NJ, Apr. 18–20), AFIPS Press, Reston, Va., 1967, pp. 483–485, when Dr. Amdahl was at International Business Machines Corporation, Sunnyvale, California." Solid-State Circuits Society Newsletter, IEEE 12.3 (2007): 19-20.

[16] Mitrion, Low power hybrid computing for efficient software acceleration,White paper,Mitrionics,2008

[17] S. Che, J. Li, J. Sheaffer, K. Skadron, and J. Lach. Accelerating compute-intensive applications with gpus and fpgas. In Symposium on Application Specific Processors, pages 101 –107, 2008.

[18] Pawlak, Z, "Elementary Rough Set Granules: Toward a Rough Set Processor," Rough-Neural computing Cognitive Technologies, Dr.. S.K. Pal et al, Springer 2004, pp.5-13.

[19] Muraszkiewicz, Mieczyslaw, and Henryk Rybinski. "Towards a parallel rough sets computer." Rough Sets, Fuzzy Sets and Knowledge Discovery. Springer London, 1994. 434-443.

[20] Lewis, M. Perkowski, and L. Jozwiak, "Learning in Hardware: Architecture and Implementation of an FPGA – Based Rough set machine," IEEE, 1999, pp.326-334.

[21] A. Kanasugi, "A Design of architecture for Rough Set Processor", New Frontiers in Artificial Intelligence, T Terano et al. ,Ed.,LNCS, vol. 2253, 2001, pp.406-410.

[22] A.Kanasugi and M. Matsumoto, "Design and Implementation of Rough Rules Generation from Logical Rules on FPGA board", *Rough Sets and Intelligent Systems Paradigms*, M. Kryszkiewicz et al, Eds,LNCS, vol. 4585, 2007, pp. 594-602.

[23] Guoqiang Sun; Xiaoming Qi; Yuanyuan Zhang, "A FPGA-based implementation of Rough Set Theory," Control and Decision Conference (CCDC), 2011 Chinese, vol., no., pp.2561-2564.

[24] Stepaniuk, Jaroslaw, Maciej Kopczynski, and Tomasz Grzes. "The First Step Toward Processor for Rough Set Methods." Fundamenta Informaticae 127.1 (2013): 429-443.

[25] Grześ, Tomasz, Maciej Kopczyński, and Jarosław Stepaniuk. "FPGA in Rough Set Based Core and Reduct Computation." Rough Sets and Knowledge Technology. Springer Berlin Heidelberg, 2013. 263-270.

[26] Kopczynski, Maciej, Tomasz Grzes, and Jaroslaw Stepaniuk. "Generating Core in Rough Set Theory: Design and Implementation on FPGA." Rough Sets and Intelligent Systems Paradigms. Springer International Publishing, 2014. 209-216.

[27] Tiwari, K. S., and A. G. Kothari. "Architecture and Implementation of Attribute Reduction Algorithm Using Binary Discernibility Matrix." Computational Intelligence and Communication Networks (CICN), 2011 International Conference on. IEEE, 2011.

[28] Tiwari, Kanchan S., Ashwin G. Kothari, and Avinash G. Keskar. "Reduct generation from binary discernibility matrix: an hardware approach." International Journal of Future Computer and Communication 1.3 (2012): 270-272.

[29] Tiwari, Kanchan, Ashwin Kothari, and Riddhi Shah. "FPGA Implementation of a Reduct Generation Algorithm based on Rough Set Theory." International Journal of Advanced Electrical and Electronics Engineering ISSN (Print) : 2278-8948, Volume-2, Issue-6, 2013 [55-61]