ROUGH SET METHODS AND HARDWARE IMPLEMENTATIONS

Maciej Kopczyński, Jarosław Stepaniuk

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: This paper describes current achievements about hardware realisation of rough sets algorithms in FPGA (**F**ield **P**rogrammable **G**ate **A**rray) logic devices. At the moment only few ideas and hardware implementations have been created. Most of the existing rough set methods implementations are software type. Software solution provides flexibility in terms of data processing and executed algorithms, but is relatively slow. Hardware implementation limits this versatility, but gives a significant increase in calculation speed.
The paper also includes brief description of current authors research on the creation of this type of implementation. The testing environment uses FPGA from *Altera* called *Cyclone II*. This is a high-capacity device providing the ability to create soft-processor core, along with modules allowing to support peripherals of the development board.

Keywords: rough sets, FPGA, programmable logic devices

1. Introduction

Professor Zdzisław Pawlak introduced rough sets assuming that objects are perceived by values of some attributes (for review see e.g., [9,10,13]). Existing implementations of the rough set methods are implemented using a high-level programming languages. This type of implementation provides the ability to comply with any of the algorithms, but the biggest issue is relatively low speed of operation.

The computer processor is a versatile system that executes an arbitrary list of compiler-generated instructions dependant on the source code created by the programmer. For this reason, processors are not optimized to perform specific actions, such as simultaneous rapid execution of elementary logical operations on large amounts of data in a set of objects.

Creating hardware implementation allow us a huge acceleration of the calculation related to the chosen topic, but the disadvantage of this approach is the limit

Zeszyty Naukowe Politechniki Białostockiej. Informatyka, vol. 8, pp. 5-17, 2011.

of the applicability of such system only to given issue. A good example of such solutions are GPU's (**G**raphics **P**rocessing **U**nit), which are optimized for parallel execution of calculations related to computer graphics. Most of the mass-produced systems are ASIC-type systems (**A**pplication **S**pecific **I**ntegrated **C**ircuit) that do not allow changes defined in their logical function. Prototype implementations of specialized processors may be implemented in programmable logic devices (e.g., CPLDs (**C**omplex **P**rogrammable **L**ogic **D**evice) and FPGAs) as sequential and combination systems.

Other possibility is a combination of both implementation techniques of the algorithms, which can take advantages of versatility known from software implementations and high-speed calculation of hardware implementations.

Rough sets theory concepts implementation in hardware device can significantly accelerate the execution time of algorithms compared to the software implementation. Logic devices can execute the whole algorithm or just the most time-consuming parts of it.

The paper is organized as follows. Section 2. contains the introduction to rough sets. In Section 3.1, Pawlak's idea of rough set processor is discussed. In Section 3.2, application of cellular networks in rough set methods is shortly recalled. In Section 3.3, some investigations of Kanasugi are presented. Section 4. includes brief description of current authors research on the creation of this type of implementation. In Conclusions, we summarize the results of the paper and we present some directions for further research.

2. Basic notions of rough set theory

Rough set theory due to Zdzisław Pawlak (1926–2006) (see e.g. [9]), is a mathematical approach to imperfect knowledge. The problem of imperfect knowledge has been tackled for a long time by philosophers, logicians and mathematicians. Recently it has become a crucial issue for computer scientists as well, particularly in the area of intelligent systems. There are many approaches to the problem of how to understand and manipulate imperfect knowledge: statistics and probability methods, fuzzy sets, rough sets (see e.g. [4], [11], [15]). One of the most successful is, no doubt, the fuzzy set theory proposed by Lotfi A. Zadeh. Statistics and probability theory can be used to build models and frameworks for particular problems. Rough set theory presents still another attempt to solve this problem. It is based on an assumption that objects are perceived by partial information about them. Due to this some objects can be indiscernible. Indiscernible objects form elementary granules. From this fact it follows that some sets can not be exactly described by available information about objects.

They are rough not crisp. Any rough set is characterized by its (lower and upper) approximations. In this section, we recall some basic definitions of rough set theory.

Let U denote a finite non-empty set of objects, to be called the universe. Further, let A denote a finite non-empty set of attributes. Every attribute $a \in A$ is a function

$$a : U \to V_a,$$

where V_a is the set of all possible values of a, to be called the domain of a. In the sequel, $a(x)$, $a \in A$ and $x \in U$, denotes the value of attribute a for object x.

Definition 1. *A pair $IS = (U, A)$ is an information system.*

Usually, the specification of an information system can be presented in tabular form.

Each subset of attributes $B \subseteq A$ determines a binary $B - indiscernibility$ relation $IND(B)$ consisting of pairs of objects indiscernible with respect to attributes from B. Thus, $IND(B) = \{(x,y) \in U \times U : \forall_{a \in B} a(x) = a(y)\}$. The relation $IND(B)$ is an equivalence relation and determines a partition of U, which is denoted by $U/IND(B)$. The set of objects indiscernible with an object $x \in U$ with respect to B in IS is denoted by $I_B(x)$ and is called $B - indiscernibility$ class. Thus, $I_B(x) = \{y \in U : (x,y) \in IND(B)\}$ and $U/IND(B) = \{I_B(x) : x \in U\}$.

Definition 2. *A pair $AS_B = (U, IND(B))$ is a standard approximation space for the information system $IS = (U, A)$, where $B \subseteq A$.*

The lower and the upper approximations of subsets of U are defined as follows.

Definition 3. *For any approximation space $AS_B = (U, IND(B))$ and any subset $X \subseteq U$, the lower and upper approximations are defined by*

$$LOW\,(AS_B, X) = \{x \in U : I_B(x) \subseteq X\},$$

$$UPP\,(AS_B, X) = \{x \in U : I_B(x) \cap X \neq \emptyset\}.$$

The lower approximation of a set X with respect to the approximation space AS_B is the set of all objects, which can be classified with certainty as objects of X with respect to AS_B. The upper approximation of a set X with respect to the approximation space AS_B is the set of all objects which can be possibly classified as objects of X with respect to AS_B.

The difference between the upper and lower approximation of a given set is called its boundary region:

$$BN\,(AS_B, X) = UPP\,(AS_B, X) - LOW\,(AS_B, X).$$

Rough set theory expresses vagueness by employing a boundary region of a set. If the boundary region of a set is empty it means that the set is crisp, otherwise the set is rough (inexact). A nonempty boundary region of a set indicates that our knowledge about the set is not sufficient to define the set precisely. One can recognize that rough set theory is, in a sense, a formalization of the idea presented by a German mathematician Gotlob Frege (1848–1925).

It is possible to express numerically the roughness $R(AS_B,X)$ of a set X with respect to B by assigning

$$R(AS_B,X) = 1 - \frac{card(LOW(AS_B,X))}{card(UPP(AS_B,X))}.$$

In this way, the value of the roughness of the set X being equal 0 means that X is crisp with respect to B, and conversely if $R(AS_B,X) > 0$ then X is rough (i.e., X is vague with respect to B). Detailed information on rough set theory is provided in [9] and [13].

3. Solutions architecture

Designed and implemented rough sets hardware devices use the PC as an external data source and an element of executing all or part of main control program. Hardware systems are used as mechanisms for performing complex calculations, so it is possible to significantly accelerate the calculation time of algorithms. This type of devices can be regarded as a kind of coprocessors. Block diagram of such solutions is shown in Fig. 1.

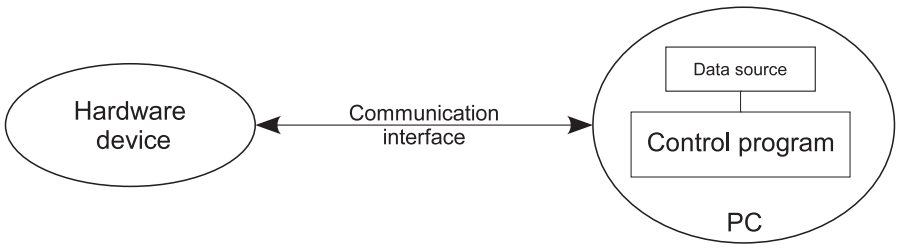

Fig. 1. Block diagram of the rough sets hardware implementation

More powerful FPGA development boards, in addition to the logic device, also contain RAM, flash memory, external flash memory connectors, communication interfaces (USB, Ethernet, etc…) and extensive microcontrollers, e.g. on the ARM

cores. Moreover, large FPGAs allow us the implementation of the soft-core processors [16] [19]. Development boards equipped within this type of devices can be used for the implementation of control software without the need for an external, large PC. Currently available versions of operating systems (e.g. Linux [18]) support most common of soft-core processors, what makes it possible to install it on this type of devices. Computing power of such processors is satisfactory for supporting these tasks. This leads to minimization of the amount of space occupied by the resulting device, which finally becomes an independent unit.

3.1 Pawlak's idea of Rough Set Processor

In [8] Pawlak presented an outline of an exemplary RSP (**R**ough **S**et **P**rocessor) structure. The organization of a simple processor is based on elementary rough set granules and dependencies between them. A simplified *RSP* is shown on Fig. 2.

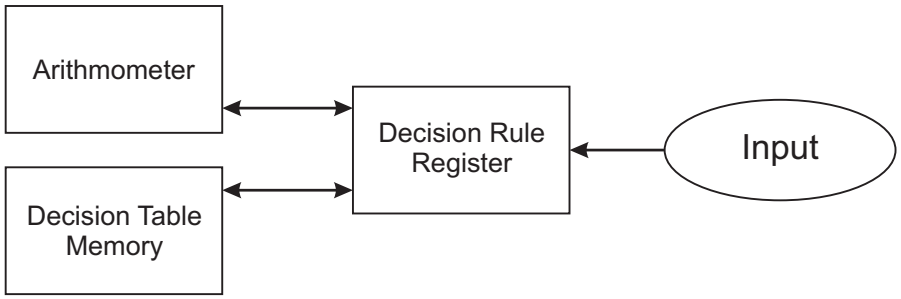

Fig. 2. Block diagram of *Rough Set Processor* [8]

RSP consists of the following units:

- *Decision Table Memory* – this unit keeps the data from the decision table. An ideal situation would be if the memory is large and fast enough to store the whole decision table,
- *Decision Rule Register* – main purpose of this unit is to generate final set of decision rules. This module cooperates with the arithmometer because of need to perform some calculations,
- *Arithmometer* – unit used to perform arithmetic operations for the rest of the modules.

The idea of *RSP* design is as follows. At the beginning, only conditions, decisions, and support of each decision rule are given. Condition and decision are parts of the decision rule. Support is the number of objects from the original decision table matching a given decision rule.

Next operation step is calculation of strength, certainty and coverage factors of each decision rule. These values will be used to find the most important decision rules.

Idea presented by Pawlak was not realized in programmable logic device.

3.2 Cellular networks

Rybinski and Muraszkiewicz in [6] created the concept of describing rough sets methods with usage of cellular networks. On this basis, the idea of the implementation of a device called *PRSComp* (**P**arallel **R**ough **S**ets **Comp**uter) was presented. This is device for parallel processing of basic rough set operations. The description of cellular networks is contained in [7]. Cellular network consists of a matrix of interconnected elements of the same type (cells, that can be treated as a simple, single processors) and a set of control registers. Block diagram of a cellular network with a set of registers is shown in Fig. 3.

The use of cellular network with rough sets is based on the transformation of the input data set to the matrix and definition of the basic operations associated with rough sets using matrix notation. In [6] the following notions are presented along with their pseudocode:

- indiscernibility relation,
- upper approximation,
- lower approximation,
- reducts calculation,
- core calculation.

Given pseudocode allows the implementation of presented matrix notation in programmable logic devices.

Paper [6] provides a basis for the development and expanding *PRSComp* device for another, more complex operations associated with rough sets and matrix notation.

Lewis, Perkowski and Jozwiak [5] described the idea of self-learning rough sets model representation in hardware device. Model is based on cellular networks by Rybinski and Muraszkiewicz. They suggested the possibility of implementing the

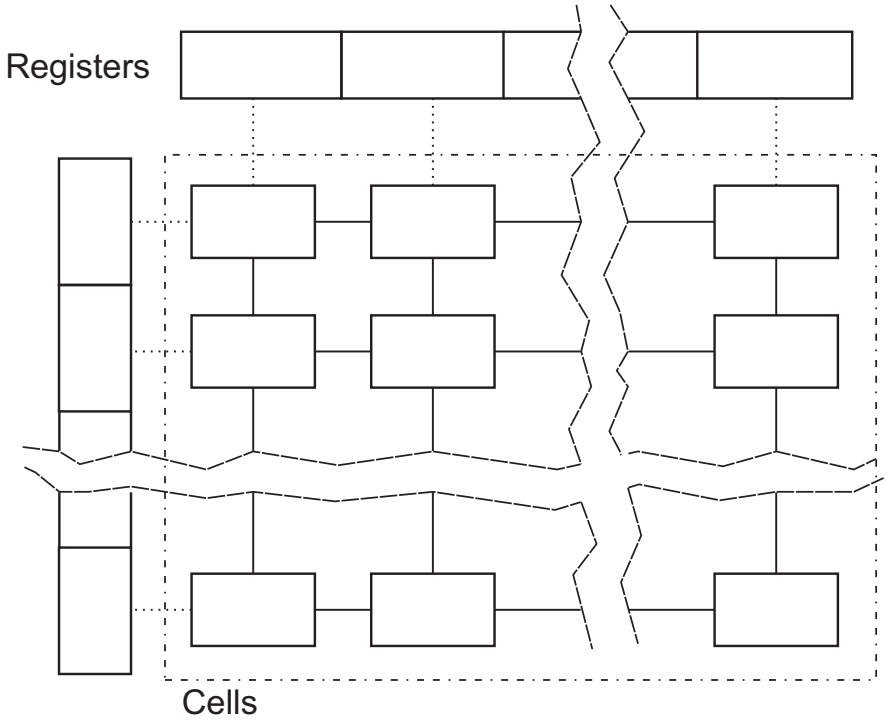

Fig. 3. Block diagram of sample cellular network

solution on *DEC-PERLE-1* board, which is the matrix consisting of 23 type 3090 FPGAs from *Xilinx*. The general principle of device operation is to perform the learning process at the higher level (software), while the later results of this process are transferred to a lower level (hardware). Description of the system working process is as follows:

1. Creation of cellular network logical structure based on an optimized decision matrix (the set of examples) and the requirements for the network construction.
2. Cellular network structure developed on the basis of data from the decision matrix is mapped to the FPGA unit, where each device performs the functions described in the *PRSComp* system. Mapping is created using standard EDA software (e.g., from *Xilinx*).
3. Device's knowledge is stored in the memory of the *DEC-PERLE-1* board as the patterns representing created cellular networks structures. Previously created cellular network patterns are multiplexed in order to choose the best one when working with different data sets. Switching scheme is supervised by an external computer with appropriate control software.
4. During the network training phase, when solving new problems, taken decisions are stored in the memory. Basing on this data the network structure can be reorganized or built completely from scratch in order to avoid the impact of the previously created pattern.

3.3 Direct solutions

Kanasugi and Yokoyama [1] developed a concept of logic device capable of minimizing the large logic functions created on the basis of discernibility matrix. System output are small logical functions representing important decision rules.

The presented system is not independent. It requires an external data source and the mechanisms for creating large logical functions from the database for correct operation. This system can be treated as a coprocessor supporting the central unit.

Block diagram of the logic device is shown in Fig. 4. The project consists of the following functional elements:

- *Core Selector* – main task of this unit is to select rows of binary decision matrix, which include the shortest logical formulas (which have the smallest number of variables with a true value).
- *Covering Unit* – goal of this unit is to check each row of a binary decision matrix and denote them as candidates to remove. Selecting the rows is based on the data prepared by the *Core Selector*.

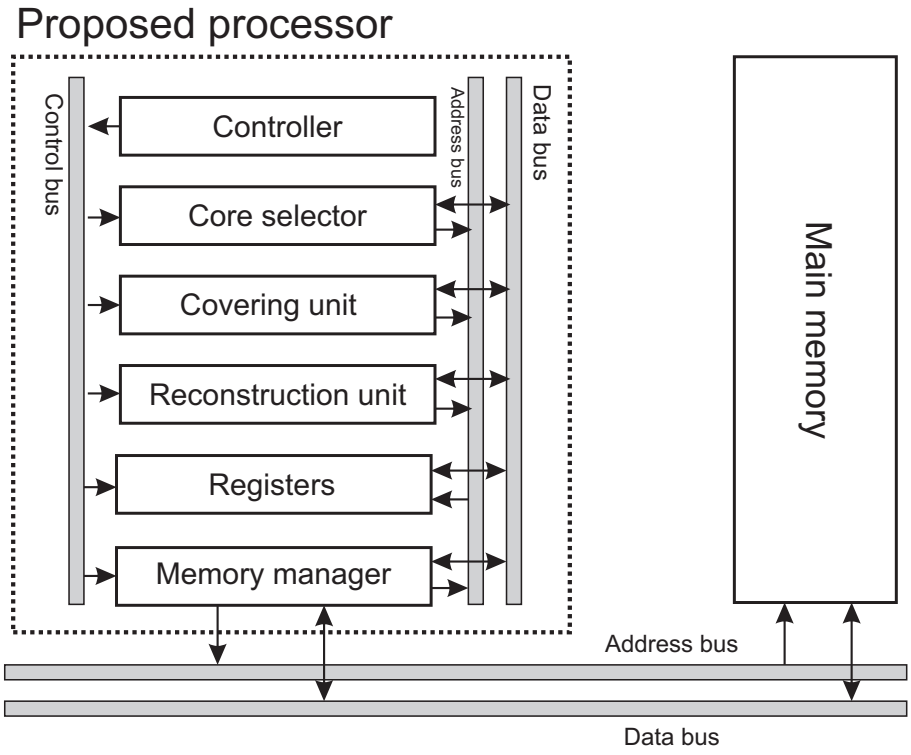

Fig. 4. Block diagram of the logic device [3]

– *Reconstruction Unit* – the role of this unit is to discover dominant variables in binary decision matrix, what helps to find most significant decision rules.

System functionality can be divided into two parts: pre-process (data preparation) and main-process (working with prepared data).
In the data pre-process two units are used: *Core Selector* and *Covering Unit*. The purpose of two mentioned units is to find the rows which contain the least amount of boolean variables (search for cores). The modules also select redundant binary decision matrix rows which can be deleted. The purpose of main-process is to review the whole pre-processed binary decision matrix and select the most important rules (terms). The idea of the algorithm implemented in *Reconstruction Unit* is to find dominant variables in pre-processed binary decision matrix and then create a new decision matrix. This matrix will contain some amount of important decision rules dependent on the algorithm parameter. It should be noted that the implemented algorithm uses approximation technique. Kanasugi has decided on this solution because of time of calculations reduction and the size of the entire system in the FPGA structure.

The solution proposed by Kanasugi and Matsumoto in [3] allowed the nearly 700 times increase in the speed of calculations in comparison to the PC. Binary decision matrix containing 128 rows of data and 2032 attributes was used during the tests. Table 1 shows the results of the experiment.

Table 1. Comparision of calculation time between hardware and software solution [3]

Device type	Speed $[MHz]$	Time $[\mu s]$
Hardware solution	50	7.18
PC	3400	72.54

4. Authors solution

The authors are working on creating a rough set hardware system, which has to be universal for any type of data. The goal of the system is to process the data in accordance with a set of rules, the rules generation from complex sets of data, fast reducts and approximations calculations and so on. Important part of the project is to create a low-level input data transformation algorithms which convert data from decision matrix or database to low-level form (boolean or binary). Inverse operation is also required — the conversion of results returned by the system to similar form found in

software implementations. These operations can be performed by software executing on the main unit. At the moment, authors are using development board *DE2-70* made by *Terasic* with FPGA *Cyclone II* (EP2C70F896) by *Altera*. Development board includes a lot of peripheral devices [17]:

- *USB Blaster* for programming (JTAG and Active Serial (AS) programming modes are supported),
- memories: 2 MB SSRAM, two 32 MB SDRAM, 8 MB Flash memory,
- SD Card socket,
- 24-bit audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector,
- 10/100 Ethernet controller,
- USB Host/Slave controller.

FPGA from *Altera* provides a support to soft-core processors. *Altera* provides the *NIOS II* soft-core processor through IP (**I**ntellectual **P**roperty) cores. Main features of the mentioned processor is [16]:

- separate instruction and data caches (512 B to 64 kB),
- optional MMU (**M**emory **M**anagement **U**nit) or MPU (**M**emory **P**rotection **U**nit),
- access to 2 GB of external address space,
- six-stage pipeline,
- single-cycle hardware multiply,
- hardware divide option,
- JTAG debug module.

Possibility of adding the MMU unit to the processor allows to run the full-featured Linux kernel, what extends the possibilities of creating stand-alone rough set device [18]. Software implemented in soft-core processor will perform control operations (e.g. data conversions, retrieving data from external sources, parallel execution synchronization), while hardware rough set units will do the calculations.

Current work emphases on implementing IO interface between external data source and the created device, creating hardware units performing basic rough sets notions such as indiscernibility relation, generating upper and lower approximations, calculating reducts and core. As soon as the device is finished, the results with comparison to the software implementation will be presented.

5. Conclusions and future research

The hardware implementation is the main direction of using scalable rough sets methods in real time solutions. Software implementations are universal, but rather slow.

Hardware realizations are deprived of this universality, however, allow us performing specific calculations in substantially shorter time.

The system with hardware implementation of rough sets methods can be used in embedded systems such as industrial controllers or as an alternative and very fast method of process control and data classification. The field of potential usage of the system can be very wide due to its versatility.

References

[1] A. Kanasugi, A. Yokoyama, A basic design for rough set processor, In The 15th Annual Conference of Japanese Society for Artificial Intelligence, 2001.

[2] A. Kanasugi, A design of architecture for rough set processor, JSAI 2001 Workshops, LNAI 2253, Springer-Verlag, 2001, pp. 406-412.

[3] A. Kanasugi, M. Matsumoto, Design and implementation of rough rules generation from logical rules on FPGA board, RSEISP 2007, LNAI 4585, Springer-Verlag, 2007, pp. 594-602.

[4] J. Koronacki, J. Cwik, Statystyczne systemy uczace sie, wydanie drugie, Exit, Warsaw, 2008, pp. 327.

[5] T. Lewis, M. Perkowski, L. Jozwiak, Learning in Hardware: Architecture and Implementation of an FPGA-Based Rough Set Machine, euromicro, vol. 1, 25th Euromicro Conference (EUROMICRO '99)-Volume 1, 1999, pp. 1326.

[6] M. Muraszkiewicz, H. Rybinski, Towards a Parallel Rough Sets Computer In: Rough Sets, Fuzzy Sets and Knowledge Discovery, Springer-Verlag, 1994, pp. 434-443.

[7] M. Muraszkiewicz, Sieci komorkowe do przetwarzania danych nienumerycznych, Prace IINTE, no. 52, 1984.

[8] Z. Pawlak, Elementary rough set granules: Toward a rough set processor. In: S. K. Pal, L. Polkowski, and A. Skowron, editors, Rough-Neurocomputing: Techniques for Computing with Words, Cognitive Technologies. Springer-Verlag, Berlin, Germany, 2004, pp. 5-14.

[9] Z. Pawlak, A. Skowron, Rudiments of rough sets. Information Sciences, 177(1) 2007, pp. 3-27.

[10] W. Pedrycz, A. Skowron, V. Kreinovich (Eds.), Handbook of Granular Computing, John Wiley & Sons, New York 2008.

[11] L. Rutkowski, Computational Intelligence, Methods and Techniques, Springer, 2008.

[12] A. Skowron, J. Stepaniuk, Tolerance Approximation Spaces, Fundamenta Informaticae, vol. 27, no. 2–3, 1996, pp. 245-253.

[13] J. Stepaniuk, Rough–Granular Computing in Knowledge Discovery and Data Mining, Springer, 2008.
[14] T. Strakowski, H. Rybinski, A Distributed Decision Rules Calculation Using Apriori Algorithm, T. Rough Sets 11, 2010, pp. 161-176.
[15] L. A. Zadeh, The role of fuzzy logic in the management of uncertainty in expert systems, Fuzzy Sets and Systems 11, 1993, pp. 199-227.
[16] Altera Corporation, www.altera.com , cited 30.11.2011.
[17] Terasic Corporation, www.terasic.com.tw , cited 30.11.2011.
[18] The Linux Kernel Archives, www.kernel.org , cited 30.11.2011.
[19] Xilinx Corporation, www.xilinx.com , cited 30.11.2011.

METODY ZBIORÓW PRZYBLIŻONYCH I IMPLEMENTACJE SPRZĘTOWE

Streszczenie Zbiory przybliżone (ang. rough sets) zostały wprowadzone przez Prof. Zdzisława Pawlaka jako narzędzie wnioskowania o pojęciach nieostrych (ang. *vague concepts*). Zarówno podstawy teoretyczne jak i zastosowania zbiorów przybliżonych zostały istotnie rozwinięte. Metody bazujące na zbiorach przybliżonych cieszą się bardzo dużym zainteresowaniem wielu środowisk na świecie.
Praca opisuje bieżące dokonania na polu implementacji sprzętowych w strukturach programowalnych FPGA (ang. **F**ield **P**rogrammable **G**ate **A**rray) metod zbiorów przybliżonych. Do tej pory stworzonych zostało zaledwie kilka takich rozwiązań. Większość istniejących implementacji metod zbiorów przybliżonych jest realizowanych programowo. Rozwiązanie programowe zapewnia uniwersalność działania pod względem przetwarzanych danych oraz wykonywanych algorytmów zapewniając jednocześnie prostotę ich modyfikacji, jednak jest relatywnie powolne. Implementacja sprzętowa ogranicza tą uniwersalność, dając jednak w zamian znaczny przyrost szybkości działania.
W pracy zawarto również krótki opis bieżących badań prowadzonych przez autorów nad stworzeniem tego typu implementacji. Do badań wykorzystywany jest układ FPGA firmy *Altera* o nazwie *Cyclone II*. Jest to układ o dużej pojemności zapewniający możliwość tworzenia procesorów typu soft-core wraz z modułami pozwalającymi na obsługę peryferiów płyty rozwojowej.

Słowa kluczowe: zbiory przybliżone, FPGA, programowalne struktury logiczne

Artykuł zrealizowano w ramach pracy badawczej S/WI/5/08.