Maciej KOPCZYŃSKI, Tomasz GRZEŚ, Jarosław STEPANIUK
WYDZIAŁ INFORMATYKI, POLITECHNIKA BIAŁOSTOCKA,
ul. Wiejska 45a, 15-351 Białystok

Realizacja algorytmu sekwencyjnego wyznaczania macierzy rozróżnialności zbiorów przybliżonych w układzie FPGA

Mgr inż. Maciej KOPCZYŃSKI

Autor od 2009 r. pracuje na Wydziale Informatyki Politechniki Białostockiej. W 2008 r. ukończył studia na kierunkach informatyka, zaś w 2010 r. elektronika i telekomunikacja. Przez wiele lat związany z przemysłem z branży ICT oraz systemów wbudowanych. Jego zainteresowania naukowe dotyczą analizy danych, zbiorów przybliżonych i wykorzystania rozwiązań sprzętowych opartych o układy FPGA w systemach informatycznych.

e-mail: m.kopczynski@pb.edu.pl

Dr inż. Tomasz GRZEŚ

Ukończył studia w Instytucie Informatyki Politechniki Białostockiej. Obronił rozprawę doktorską w 2010 r. na Wydziale Informatyki Politechniki Białostockiej. Zajmował się metodami minimalizacji poboru energii układów cyfrowych. Obecnie pracuje nad sprzętowymi implementacjami metod zbiorów przybliżonych w układach programowalnych FPGA.

e-mail: t.grzes@pb.edu.pl

Streszczenie

W niniejszym artykule przedstawiono implementację sprzętową algorytmu stosowanego w obliczeniach związanych ze zbiorami przybliżonymi służącego do wyznaczania macierzy rozróżnialności. Istniejące dotychczas rozwiązania implementowały algorytm w językach programowania wysokiego poziomu. W wyniku prac badawczych stworzono i opisano w języku VHDL układ kombinacyjny realizujący równoważne obliczenia. Przeprowadzono badania porównawcze pod względem czasu potrzebnego do zakończenia obliczeń. Uzyskane wyniki pokazują ogromne przyspieszenie układu sprzętowego w porównaniu do implementacji programowej.

Słowa kluczowe: zbiory przybliżone, układy programowalne, FPGA.

Realization of a sequential algorithm related to rough sets methodology in FPGA

Abstract

In this paper the authors present an example of sequential software algorithm implementation as a hardware unit using VHDL in FPGA programmable logic structure. The converted algorithm is one of the principal operations in the rough sets theory – discernibility matrix calculation. Rough sets methods are used in data analysis, knowledge discovery and datasets attributes downsizing. At present there are no complete hardware implementations of rough sets methods. The existing solutions are only software implementations which need huge amount of time for processing big datasets. The authors created hardware implementation of such an algorithm as a pure combinational unit described in the VHDL language. Software implementation was also created to compare processing times between two solutions. The obtained results show that the usage of a hardware processing unit gives huge acceleration in terms of the time needed to finish creating a discernibility matrix. The FPGA structure utilization focused on LEs (Logical Elements) and pins usage was also examined. The first section of the paper is an introduction to rough sets and FPGA structures. In the second section there are presented the example of entry dataset and the calculated discernibility matrix. This section also includes description of the algorithm for creating a discernibility matrix as well as the proposed hardware solution. The third section presents the experimental results for the processing time and FPGA structure utilization. The last section focuses on conclusions and plans for future research.

Keywords: rough sets, programmable logic structures, FPGA.

Prof. dr hab. Jarosław STEPANIUK

Profesor zwyczajny na Wydziale Informatyki Politechniki Białostockiej, kierownik Katedry Systemów Informacyjnych i Sieci Komputerowych; Zainteresowania naukowe: zbiory przybliżone, eksploracja danych, obliczenia granularne, sztuczna inteligencja.

e-mail: j.stepaniuk@pb.edu.pl

1. Wstęp

Teoria zbiorów przybliżonych została zainicjowana w latach osiemdziesiątych XX-wieku przez prof. Zdzisława Pawlaka. Do chwili obecnej metodologia ta zdobyła ogromną popularność w świecie naukowym i na gruncie zastosowań praktycznych. Teoria zbiorów przybliżonych stanowi uzupełnienie do teorii zbiorów rozmytych i rachunku prawdopodobieństwa. Wymienione trzy podejścia stanowią wspólnie znakomitą metodologię do pozyski-wania wiedzy z niepełnych, niepewnych i nieprecyzyjnych danych [10]. Metody zbiorów przybliżonych są wykorzystywane jako narzędzie do pozyskiwania wiedzy z baz danych oraz eksploracji danych, jak również zmniejszania liczb kolumn bez straty istotnej informacji w bazach danych. Są one użyteczne w zakresie wydo-bywania istotnych wzorców opisujących pojęcia (zbiory obiektów). Metody zbiorów przybliżonych są skuteczne w generowaniu reguł klasyfikujących nowe obiekty. Ze względu na swoją uniwersalność, są one szeroko wykorzystywane w różnych dziedzinach życia, m.in. w medycynie, farmakologii, bankowości, badaniach rynku, badaniach giełdowych, kontroli i sterowaniu urządzeń oraz procesów przemysłowych, przetwarzaniu obrazów i dźwięku (w tym sygnałów mowy) oraz analizie zawartości dokumentów sieci WWW. Zbiory przybliżone pozwalają na skuteczne wnioskowanie w warunkach często występującej w świecie rzeczywistym nie-pełności informacji [8].

Układy programowalne FPGA składają się z wielu elementów logicznych, których połączenia są programowalne co pozwala na realizację szerokiej klasy układów kombinacyjnych lub sekwencyjnych [1]. Dodatkowo zawierają one w swojej strukturze między innymi pamięć RAM o dowolnej organizacji, układy arytmetyczne (np. układy mnożące) i inne bloki funkcjonalne stosowane przy budowie nowoczesnych systemów wbudowanych. Bardzo dynamiczny rozwój technologii układów FPGA na przestrzeni ostatniej dekady stworzył nowe możliwości zastosowania ich w różnych działach informatyki. Przyniosły one wiele zastosowań praktycznych z wykorzystaniem FPGA. Dla układów FPGA istnieją bardzo rozbudowane biblioteki IP-cores realizujące zaawansowane funkcje, w tym również rdzenie mikroprocesorów. Dzięki temu w pojedynczym układzie scalonym można zawrzeć zarówno blok realizujący metody zbiorów przybliżonych, jak i mikroprocesor, tworząc zaawansowany system przetwarzania danych.

Większość istniejących implementacji metod zbiorów przybliżonych jest realizowana z wykorzystaniem wysokopoziomowych języków programowania. Implementacja tego typu zapewnia możliwość wykonania dowolnych algorytmów oraz operacji, jednak jest to okupione relatywnie małą szybkością działania. Z tego względu metody zbiorów przybliżonych nie są szeroko wykorzystywane w praktycznych systemach analizy bardzo dużych zbiorów danych (Big Data). Wykonanie w strukturze FPGA układów realizujących metody związane z teorią zbiorów przybliżo-

nych i wnioskowanie bazujące na tej teorii pozwala na znaczne przyspieszenie czasu wykonania algorytmów w porównaniu do realizacji programowych istniejących algorytmów.

Nowatorski charakter proponowanego rozwiązania wynika z braku istniejących kompleksowych i w pełni funkcjonalnych rozwiązań tego typu [4, 5]. Rozwiązanie zaproponowane w ramach niniejszego artykułu jest przykładem przeniesienia programowego algorytmu sekwencyjnego o wielomianowej złożoności obliczeniowej na grunt programowalnej struktury logicznej typu FPGA. Algorytm ten dotyczy obliczenia macierzy rozróżnialności – jednej z podstawowych konstrukcji w teorii zbiorów przybliżonych będącej punktem wyjścia do obliczenia innych elementów (np. rdzenia czy reduktu).

2. Algorytm wyznaczania macierzy rozróżnialności

W rozdziale tym zaprezentowane są przykładowe dane wejściowe i stworzona na ich podstawie macierz rozróżnialności Rozdział zawiera również opis algorytmu wyznaczania macierzy rozróżnialności i koncepcję implementacji sprzętowej realizującej tą samą operację w strukturze programowalnej.

Wejściem algorytmu jest tablica decyzyjna. Jej przykład pokazany jest w tab. 1.

Tab. 1. Przykład tablicy decyzyjnej
Tab. 1. Example of a decision table

Nr	*Aura*	*Temperatura*	*Wilgotność*	*Wiatr*	*Zawody*
x1	słoneczna	ciepło	duża	silny	Nie
x2	deszczowa	umiarkowana	duża	słaby	Tak
x3	słoneczna	ciepło	duża	silny	Nie
x4	pochmurna	ciepło	normalna	słaby	Tak
x5	deszczowa	umiarkowana	duża	słaby	Tak
x6	pochmurna	zimno	normalna	silny	Nie

Jest to dwuwymiarowa tabela, której wiersze opisują poszczególne obiekty (U), zaś kolumny są to atrybuty. Ich zbiór dzieli się na dwie części: warunkowe (A) i decyzyjne (d). Do zbioru atrybutów warunkowych mogą należeć *aura, temperatura, wilgotność, wiatr*, zaś zbiór atrybutów decyzyjnych może być jednoelementowy i składać się z kolumny *zawody*. Atrybuty decyzyjne określają klasę obiektu, zaś warunkowe definiują cechy brane pod uwagę w procesie rozpoznawania obiektu.

Wyjściem algorytmu jest macierz rozróżnialności. Jej przykład pokazany jest w tab. 2. W celu zwiększenia przejrzystości zapisu poszczególnym atrybutom zostały przyporządkowane skróty: aura (a), temperatura (t), wilgotność (h), wiatr (w).

Tab. 2. Macierz rozróżnialności wyznaczona dla tablicy decyzyjnej z tab. 1
Tab. 2. Discernibility matrix for the decision table shown in tab. 1

	x1	*x2*	*x3*	*x4*	*x5*	*x6*
x1	∅	{a,t,w}	∅	{a,h,w}	{a,t,w}	∅
x2	{a,t,w}	∅	{a,t,w}	∅	∅	{a,t,h,w}
x3	∅	{a,t,w}	∅	{a,h,w}	{a,t,w}	∅
x4	{a,h,w}	∅	{a,h,w}	∅	∅	{a,t,w}
x5	{a,t,w}	∅	{a,t,w}	∅	∅	{a,t,h,w}
x6	∅	{a,t,h,w}	∅	{a,t,w}	{a,t,h,w}	∅

Macierz rozróżnialności jest tablicą kwadratową dwuwymiarową o rozmiarze n równym ilości obiektów w tablicy decyzyjnej. Przedstawia ona różnice pomiędzy poszczególnymi obiektami w odniesieniu do atrybutów warunkowych. Macierze rozróżnialności są istotnym elementem w teorii zbiorów przybliżonych, gdyż stanowią punkt wyjścia do obliczania innych relacji w metodach zbiorów przybliżonych (np. rdzenie).

Poniżej przedstawiony jest pseudokod algorytmu wyznaczania macierzy rozróżnialności:

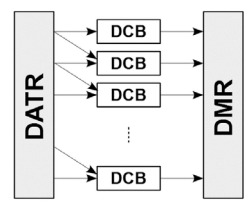
WEJŚCIE: tablica decyzyjna *TD* o *n* wierszach i *k* kolumnach
WYJŚCIE: macierz rozróżnialności *MR* o rozmiarze *n* wierszy i *n* kolumn

ALGORYTM:
```
1  for (x ∈ U ) do
2    for (y ∈ U ) do
3      set := ∅
4      for (a ∈ A ) do
5        if a(x) ≠ a(y) and d(x) ≠ d(y) then
6          set := set ∪ {a}
7        end if
8      end for
9      MR(x,y) := set
10   end for
11 end for
```

Operacja $a(x)$ oznacza pobranie wartości atrybutu a dla obiektu x. Algorytm rozpoczyna się pętlą wybierającą pierwszy obiekt do porównania (wiersz 1). Kolejna pętla przechodzi po pozostałych obiektach należących do tablicy decyzyjnej (wiersz 2). Na początku drugiej pętli inicjalizowany jest zbiór atrybutów *set* rozróżniających obiekty x i y (wiersz 3). Trzecia pętla (wiersz 4) iteruje po wszystkich atrybutach warunkowych. Instrukcja warunkowa (wiersz 5) sprawdza, czy dwa obiekty x i y należą do różnych klas decyzyjnych (innymi słowy – czy wartość atrybutu decyzyjnego d jest różna), jak również czy obiekty x i y różnią się pod względem wartości na atrybucie a. Jeśli warunek jest spełniony, to zbiór *set* jest powiększany o atrybut a (wiersz 6). Po zakończeniu pętli iterującej po atrybutach warunkowych, zbiór *set* jest wstawiany do komórki macierzy rozróżnialności *MR* leżącej na przecięciu wiersza z obiektem x i kolumny z obiektem y (wiersz 9). Jak łatwo zauważyć, na przekątnej macierzy rozróżnialności występują zbiory puste oraz jest macierz symetryczna względem głównej przekątnej.

3. Sprzętowa realizacja algorytmu

Równoważne rozwiązanie sprzętowe zostało zrealizowana jako układ w pełni kombinacyjny. Implementacja zrealizowana została z wykorzystaniem języka VHDL w środowisku *Quartus II* 13.1 firmy *Altera*. Schemat blokowy rozwiązania przedstawiony jest na rys. 1.



Rys. 1. Schemat blokowy układu obliczającego macierz rozróżnialności
Fig. 1. Block diagram of discernibility matrix calculation

Poszczególne elementy wchodzące w skład modułu są następujące:

- **DATR** (*DATa Register*) – rejestr wejściowy zawierający tablicę decyzyjną do przetworzenia. Rozmiar rejestru wejściowego R_{DATR} jest równy

$$R_{DATR} = k \cdot n \cdot s, \tag{1}$$

gdzie: n – liczba obiektów, k – liczba wszystkich atrybutów, s – szerokość w bitach jednego atrybutu.

- **DCB** (*Discernibility Comparator Block*) – blok, którego zadaniem jest porównanie między sobą dwóch obiektów. Każdy blok posiada 2 wejścia o szerokości $k \cdot s$ oraz jedno wyjście o szerokości jednego bita, które informuje o występowaniu różnicy po porównaniu dwóch obiektów między sobą. Blok ten jest powielony wielokrotnie i połączony z reprezentacjami bitowymi poszczególnych obiektów zawartych w tablicy decyzyjnej (czyli w rejestrze **DATR**)

- **DMR** (*Discernibility Matrix Register*) – rejestr przechowujący dane przetworzone przez bloki **DCB** – czyli macierz rozróżnialności. Rejestr przechowuje dolny trójkąt macierzy rozróżnialności bez przekątnej. Rozmiar R_{DMR} rejestru jest równy:

$$R_{DMR} = \frac{n \cdot (n-1)}{2} \cdot (k-1) . \qquad (2)$$

Powielenie bloków **DCB** pozwoliło na stworzenie układu sprzętowego w pełni kombinacyjnego, dzięki czemu obliczenie macierzy rozróżnialności nie wymaga stosowania zegara taktującego. Oczywiście sygnału taktującego nie da się uniknąć ze względu na synchronizację z pozostałymi elementami systemu, jednak sam proces obliczania macierzy rozróżnialności działa natychmiast po podaniu danych do rejestru **DATR**. Czas potrzeby na otrzymanie stabilnego wyniku na wyjściach jest zależny jedynie od czasu propagacji sygnału w blokach logicznych układu FPGA, który z reguły nie przekracza 10 ns dla pojedynczej komórki LUT (ang. *Look-up Table*) w przypadku większości dostępnych na rynku modeli.

Przed przetworzeniem danych przez moduł sprzętowy wymagana jest konwersja tablicy decyzyjnej na postać binarną, dostosowaną do wymogów wejścia układu.

4. Wyniki badań eksperymentalnych

W celu porównania efektywności rozwiązania sprzętowego, stworzona została równoważna implementacja algorytmu w języku C. Głównym powodem wyboru tego języka był deterministyczny czas wykonania, duża elastyczność i kontrola w tworzeniu implementacji, względna prostota tworzenia niskopoziomowych kanałów komunikacyjnych, jak również przyszłe plany związane z przeniesieniem oprogramowania sterującego na mikroprocesor, dzięki czemu będzie mogło powstać rozwiązanie do przetwarzania i analizy danych gotowe do zastosowania w systemach wbudowanych.

Wyniki dla implementacji programowej zostały uzyskane na komputerze PC wyposażonym w 8GB pamięci RAM i 4-rdzeniowy procesor *Intel* i7 3632QM taktowany maksymalnym zegarem 3,2 GHz w trybie *Turbo*. Komputer posiada system operacyjny *Windows* 7 *Professional*. Kod źródłowy skompilowany został za pomocą kompilatora GNU GCC w wersji 4.8.1. Podane czasy dla realizacji programowej są czasami uśrednionymi uzyskanymi z wykonania danego algorytmu 10 000 razy.

Implementacja sprzętowa stworzona w języku VHDL wykorzystywała moduł w konfiguracji pozwalającej na jednorazowe przetworzenie całej wejściowej tablicy decyzyjnej. Do kompilacji, syntezy i symulacji weryfikacyjnej wykorzystane zostało środowisko *Quartus* II 13.1. Zsyntezowany kod załadowany został do płyty uruchomieniowej firmy *TeraSIC DE*-3 wyposażonej w układ FPGA firmy *Altera* z serii *Stratix III* (EP3SL150F1152C2N). Pomiary czasu trwania obliczeń wykonane zostały z użyciem oscyloskopu firmy *Tektronix* model TDS3052B (pasmo 500 MHz, próbkowanie 5 GS/s).

Do badań wykorzystana została baza danych zawierająca opis osób chorych na cukrzycę i jedno z możliwych powikłań [9]. Baza składa się ze 107 obiektów i 13 atrybutów (jeden z nich wybrany został jako decyzyjny). Badania przeprowadzone zostały na pełnej bazie, jak również na jej podzbiorach zawierających mniejszą liczbę obiektów.

W tab. 3 przedstawione są wyniki czasów wykonania dla implementacji sprzętowej oraz programowej.

Tab. 3. Porównanie czasów wykonania dla implementacji sprzętowej oraz programowej obliczania macierzy rozróżnialności
Tab. 3. Comparison of the processing time for software and hardware implementation for discernibility matrix calculation

Liczba obiektów	**Program** – t_s	**Sprzęt** – t_h	$\frac{t_s}{t_h}$
–	**[µs]**	**[µs]**	–
30	410	0,0050	82 000
45	950	0,0084	113 095
60	1 670	0,0114	146 491
90	3 920	0,0192	204 166
107	5 780	0,0246	234 959

Tab. 4 zawiera dane dotyczące zajętości struktury programowalnej FPGA w zależności od rozmiaru wejściowej tablicy decyzyjnej. Zajętość wyrażona jest w ilości elementów logicznych *LE* (ang. *Logical Element*) dostępnych w układzie FPGA.

Tab. 4. Zajętość struktury programowalnej FPGA dla układu sprzętowego obliczania macierzy rozróżnialności
Tab. 4. Programmable logic utilization for the hardware discernibility matrix calculation unit

Liczba obiektów	**Liczba LE**	**Liczba wyprowadzeń**
30	5 655	6 013
45	12 870	13 063
60	23 010	22 813
90	52 065	50 413
107	73 723	70 847

Zaprezentowane w Tab. 3 wyniki pokazują ogromne przyspieszenie związane z czasem przetwarzania danych w porównaniu do rozwiązania programowego. Implementacja sprzętowa jest co najmniej o 4 rzędy wartości szybsza niż równoważne rozwiązanie programowe. Jak łatwo zauważyć, układ jest tym szybszy, im większy zbiór danych jest jednocześnie przetwarzany.

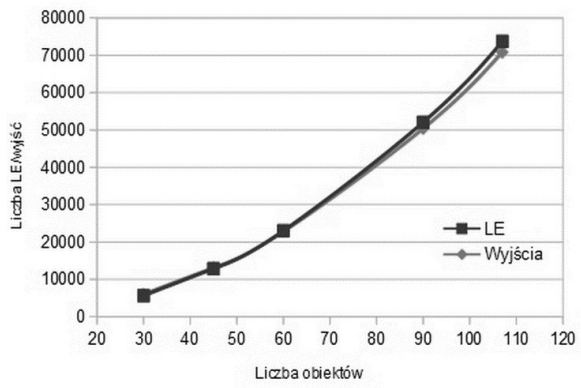
Wyznaczenie złożoności obliczeniowej algorytmu generowania macierzy rozróżnialności wymaga postawienia następujących założeń:
- operacją elementarną jest porównanie wartości pojedynczego atrybutu dla dwóch obiektów lub pobranie jednego atrybutu z kontenera danych (np. lista dynamiczna)
- moduł sprzętowy jest na tyle duży, że może przetworzyć jednocześnie całą tablicę decyzyjną.

Niech k oznacza liczbę atrybutów, zaś n określa liczbę obiektów. W przypadku rozwiązania programowego, złożoność obliczeniowa algorytmu generowania macierzy rozróżnialności na podstawie tablicy decyzyjnej o rozmiarze $n \times k$ wynosi $\Theta(n^2k)$. W przypadku zaprezentowanego rozwiązania sprzętowego zrealizowanego jako układ kombinacyjny złożoność obliczeniowa jest równa $\Theta(1)$.

Oczywiste jest, że w przypadku, gdy zbiór danych będzie większy, niż maksymalny rozmiar modułu sprzętowego ograniczony przez pojemność struktury programowalnej FPGA, to złożoność obliczeniowa rozwiązania sprzętowego zbliży się do rozwiązania programowego. Jest to związane z potrzebą podziału wejściowego zbioru danych na mniejsze podzbiory, pasujące pod względem struktury do pojedynczej sprzętowej jednostki przetwarzającej. Przesuwanie takiego okna danych po tablicy wejściowej będzie powodowało przybliżenie kosztów obliczeniowych implementacji sprzętowej do implementacji programowej. Jednak mimo tego,

rozwiązanie sprzętowe będzie nadal szybsze, głównie ze względu na realizację w jednym cyklu zegara synchronizującego przepływ danych wielu operacji porównania na długich ciągach bitów. W przypadku procesora komputera PC, wykonanie tych samych działań wymaga wielu tysięcy cykli zegara.

Analizując dane zamieszczone w Tab. 4 można zauważyć, że układ kombinacyjny realizujący algorytm wyliczania macierzy rozróżnialności wykorzystuje dużą ilość zasobów układu programowalnego FPGA. Rys. 2 przedstawia zależność zajętości struktury FPGA w zależności od liczby obiektów w wejściowej tablicy decyzyjnej.



Rys. 2. Zajętość struktury programowalnej FPGA w zależności od ilości obiektów
Fig. 2. Resources utilization of the FPGA chip vs. the number of objects

Można zauważyć, że zajętość bloków LE oraz wykorzystanie wyprowadzeń rośnie w sposób wielomianowy w zależności od ilości obiektów wejściowych. W zawiązku z tym w ostatecznym rozwiązaniu wymagane jest określenie maksymalnego rozmiaru sprzętowego bloku obliczającego macierz rozróżnialności biorąc pod uwagę docelowy układ FPGA oraz inne układy sprzętowe występujące w tworzonym systemie w tej samej strukturze FPGA.

Potencjalnie problematyczna wydaje się również ilość wyprowadzeń układu wyjściowego. Jednym z rozwiązań tego problemu jest zastosowanie szeregowych lub szeregowo-równoległych interfejsów transmisji danych, bądź dołączenie do modułu sprzętowego obliczania macierzy rozróżnialności innych układów sprzętowych (np. obliczania rdzenia), których liczba wyjść jest wielokrotnie mniejsza (np. dla rdzenia liczba wyjść może być równa liczbie atrybutów).

5. Wnioski

Implementacje sprzętowe tradycyjnych sekwencyjnych algorytmów realizowanych w językach programowania wysokiego poziomu są jedną z metod znacznego skrócenia czasu ich wykonania. Kwestia ta jest szczególnie istotna w aplikacjach realizowanych w systemach wbudowanych, gdzie zasoby obliczeniowe i pamięciowe są znacznie ograniczone w stosunku do typowych komputerów PC. Rozwiązania tego typu sprawdzą się również w systemach, w których bardzo szybki czas reakcji odgrywa kluczową role, a zwłoka w przekazaniu wyników analizy danych wejściowych jest nieakceptowalna.

Rozwiązania tego typu mają bardzo duży potencjał wykorzystania komercyjnego, szczególnie w aspekcie szybkiego przetwarzania danych. W dzisiejszych czasach ilość danych do przetwarzania rośnie dużo szybciej niż moc obliczeniowa komputerów, które mogłyby te dane w akceptowalnym czasie przetworzyć.

Częstym problemem w konstruowaniu tego typu implementacji jest brak spójnej metodologii konwersji sekwencyjnych algorytmów programowych na alternatywne rozwiązania sprzętowe z maksymalizacją części kombinacyjnej. Kolejna kwestia jest związana z brakiem możliwości, bądź też dużym utrudnieniem w zakresie identyfikacji elementarnych operacji, których realizacja sprzętowa pozwoli uzyskiwać bardzo duże przyspieszenia w stosunku do konwencjonalnych realizacji programowych.

Dalsze badania autorów będą dotyczyły algorytmów dekompozycji wejściowych tablic decyzyjnych dla potrzeb sprzętowych realizacji metod zbiorów przybliżonych (między innymi zaprezentowanego modułu sprzętowego obliczania macierzy rozróżnialności). Badania będą związane również z implementacją innych metod teorii zbiorów przybliżonych.

Badania finansowane są z pracy statutowej nr S/WI/3/2013 Wydziału Informatyki Politechniki Białostockiej oraz z grantu Narodowego Centrum Nauki nr 2012/07/B/ST6/01504.

6. Literatura

[1] Athanas P., Pnevmatikatos D., Sklavos N. (Eds.): Embedded Systems Design with FPGAs, Springer, 2013.

[2] Grześ T., Kopczyński M., Stepaniuk J.: FPGA in rough set based core and reduct computation, Lecture Notes in Computer Science Vol.8171: Lecture Notes in Articial Intelligence, Rough sets and knowledge technology: 8th International Conference: RSKT2013, eds. Pawan Lingras, Marcin Wolski, Chris Cornelis, Sushmita Mitra, Piotr Wasilewski, Berlin, Springer-Verlag, s. 263-270, 2013.

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

[4] Kopczyński M., Stepaniuk J.: Rough set methods and hardware implementations, Zeszyty Naukowe Politechniki Białostockiej. Informatyka Zeszyt 8, s. 5-18, 2011.

[5] Kopczyński M., Stepaniuk J.: Hardware Implementations of Rough Set Methods in Programmable Logic Devices, Rough Sets and Intelligent Systems – Professor Zdzisław Pawlak in Memoriam, eds. Andrzej Skowron, Zbigniew Suraj, Intelligent Systems Reference Library 43, Heidelberg, Springer, s. 309-321, 2013.

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

[7] Pawlak Z.: 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, pp. 5-14, 2004.

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

[9] Stepaniuk J.: Knowledge discovery by application of rough set models. In: L. Polkowski, S. Tsumoto, T.Y. Lin (Eds.), Rough Set Methods and Applications. New Developments in Knowledge Discovery in Information Systems, Physica-Verlag, Heidelberg, 137-233, 2000.

[10] Stepaniuk J.: Rough-Granular Computing in Knowledge Discovery and Data Mining, Springer, 2008.

[11] Stepaniuk J., Kopczyński M., Grześ T.: The First Step Toward Processor for Rough Set Methods, Fundamenta Informaticae Vol. 127, s. 429-443, 2013.

otrzymano / received: 06.02.2014
przyjęto do druku / accepted: 01.04.2014 *artykuł recenzowany / revised paper*