Łukasz IWANECKI [2], Sebastian KORYCIAK [1,2], Agnieszka DĄBROWSKA-BORUCH [1,2], Kazimierz WIATR [1,2]
[1] AGH AKADEMIA GÓRNICZO-HUTNICZA W KRAKOWIE, ACK CYFRONET AGH, ul. Nawojki 11, 30-950 Kraków
[2] AGH AKADEMIA GÓRNICZO-HUTNICZA W KRAKOWIE, KATEDRA ELEKTRONIKI, Al. Mickiewicza 30, 30-059 Kraków

Wykorzystanie akceleracji sprzętowej przy implementacji metryk podobieństwa tekstów

Inż. Łukasz IWANECKI

Ukończył studia pierwszego stopnia na AGH (2013), wydział Informatyki, Elektroniki i Telekomunikacji na kierunku Elektronika. Obecnie kończy studia magisterskie o specjalności Aparatura Elektroniczna. Jego zainteresowania to sprzętowa implementacja algorytmów, budowa interfejsów człowiek – maszyna przy pomocy mikrokontrolerów.

e-mail: iwanecki@gmail.com

Dr inż. Agnieszka DĄBROWSKA - BORUCH

Absolwentka kierunku Elektronika i Telekomunikacja na Wydziale EAIiE AGH (2002), dr nauk technicznych (2007). Obecnie jest adiunktem w Katedrze Elektroniki AGH oraz członkiem Zespołu Akceleracji Obliczeń ACK Cyfronet AGH. Jej zainteresowania naukowe to kompresja obrazu, systemy czasu rzeczywistego, układy programowalne oraz rekonfigurowalne.

e-mail: adabrow@agh.edu.pl

Mgr inż. Sebastian KORYCIAK

Ukończył studia na AGH (2011), wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obecnie jest doktorantem na wydziale Informatyki, Elektroniki i Telekomunikacji, asystentem w Katedrze Elektroniki AGH oraz członkiem Zespołu Akceleracji Obliczeń ACK Cyfronet AGH. Jego zainteresowania to implementacja algorytmów kompresji obrazów i sieci neuronowych przy pomocy układów programowalnych oraz systemy heterogeniczne.

e-mail: koryciak@agh.edu.pl

Prof. dr hab. inż. Kazimierz WIATR

Studia AGH Kraków (1980), doktor nauk technicznych (1987), doktor habilitowany (1999) i profesor (2002). Profesor zwyczajny w Akademii Górniczo - Hutniczej oraz dyrektor ACK Cyfronet AGH. Prowadzone prace badawcze dotyczą komputerowego sterowania procesami, systemów wizyjnych, systemów wieloprocesorowych, układów programowalnych, rekonfigurowalnych systemów obliczeniowych i sprzętowych metod akceleracji obliczeń.

e-mail: wiatr@agh.edu.pl

Streszczenie

Artykuł opisuje badania na temat klasyfikatorów tekstów. Zadanie polegało na zaprojektowaniu akceleratora sprzętowego, który przyspieszyłby proces klasyfikacji tekstów pod względem znaczeniowym. Projekt został podzielony na dwie części. Celem części pierwszej było zaproponowanie sprzętowej implementacji algorytmu realizującego metrykę do obliczania podobieństwa dokumentów. W drugiej części zaprojektowany został cały systemem akceleratora sprzętowego. Kolejnym etapem projektowym jest integracja modelu metryki z system akceleracji.

Słowa kluczowe: Akceleracja sprzętowa, FPGA, ARM, klasyfikacja tekstu.

The use of a hardware accelerator for implementation of text resemblance metrics

Abstract

The aim of this project is to propose a hardware accelerating system to improve the text categorization process. Text categorization is a task of categorizing electronic documents into the predefined groups, based on the content. This process is complex and requires a high performance computing system and a big number of comparisons. In this document, there is suggested a method to improve the text categorization using the FPGA technology. The main disadvantage of common processing systems is that they are single-threaded – it is possible to execute only one instruction per a single time unit. The FPGA technology improves concurrence. In this case, hundreds of big numbers may be compared in one clock cycle. The whole project is divided into two independent parts. Firstly, a hardware model of the required metrics is implemented. There are two useful metrics to compute a distance between two texts. Both of them are shown as equations (1) and (2). These formulas are similar to each other and the only difference is the denominator. This part results in two hardware models of the presented metrics. The main purpose of the second part of the project is to design a hardware accelerating system. The system is based on a Xilinx Zynq device. It consists of a Cortex-A9 ARM processor, a DMA controller and a dedicated IP Core with the accelerator. The block diagram of the system is presented in Fig.4. The DMA controller provides duplex transmission from the DDR3 memory to the accelerating unit omitting a CPU. The project is still in development. The last step is to integrate the hardware metrics model with the accelerating system.

Keywords: hardware acceleration, FPGA, ARM, text classification.

1. Wstęp

Mowa ludzka to uniwersalny system do przekazywania wiedzy. Umiejętność posługiwania się językiem umożliwia nam szybką ewolucję. Aktualny stan rozwoju elektroniki umożliwia magazynowania i przetwarzanie dużej ilości informacji. Problem automatycznej kategoryzacji dokumentów tekstowych wywodzi się z pogranicza przetwarzania języka naturalnego i uczenia maszynowego. Jej celem jest przyporządkowanie dokumentu do jednej lub wielu zdefiniowanych klas, które zostały określone przez nauczyciela, czyli zbiór uczący. Proces ten zostaje wykonany przy pomocy odpowiednich metryk.

2. Opis metryk podobieństwa tekstów

W modelu boolowskim opisu metryk, na którym jest oparty projekt, można wyróżnić dwie metryki [2, 3]. Są to metryki podobieństwa (ang. resemblance) oraz zawierania (ang. containment). Metryka podobieństwa określa, jak bardzo dwa pliki, A i B, są do siebie podobne w oparciu o odpowiadające im zbiory n-gramów.

$$r(A,B) = \frac{\left|S(A,n) \cap S(B,n)\right|}{\left|S(A,n) \cup S(B,n)\right|} \tag{1}$$

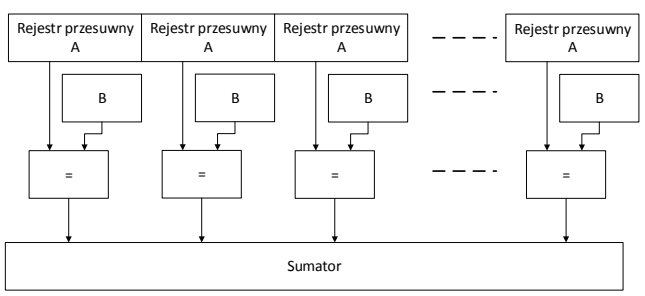
We wzorze tym $S(A,n)$ jest zbiorem n-gramów dla dokumentu A, natomiast $S(B,n)$ jest zbiorem n-gramów dla dokumentu B. Metryka podobieństwa zmienia się w zakresie od 0 do 1, gdzie wartość 1 oznacza, że oba dokumenty posiadają te same zbiory n-gramów (są statystycznie takie same), natomiast 0, gdy nie mają żadnej części wspólnej. Współczynnik zawierania określa, w jak dużym stopniu dokument A jest zawarty gdzieś w dokumencie B.

$$c(A,B) = \frac{\left|S(A,n) \cap S(B,n)\right|}{\left|S(A,n)\right|} \tag{2}$$

Parametr ten przyjmuje wartości z przedziału od 0 do 1, gdzie wartość 1 oznacza, że zbiór n-gramów dokumentu *A* całkowicie zawiera się w zbiorze n-gramów dokumentu *B*. Warto zauważyć, iż oba wzory różnią się jedynie mianownikiem. Ponadto w obu wzorach najbardziej krytyczny obliczeniowo jest sam licznik ułamka. Należy znaleźć liczbę takich samych elementów w dwóch zbiorach. Są to zazwyczaj zbiory o bardzo dużej liczbie elementów (rzędu dziesiątek tysięcy). Ponadto w ich skład wchodzą duże liczby, przynajmniej 32-bitowe. Porównanie ze sobą w krótkim czasie tak wielu dużych liczb, metodą „każda z każdą", jest niewątpliwie procesem złożonym obliczeniowo. Podczas trwania projektu podjęto próbę implementacji podanych metryk w strukturze FPGA[1]. Zastosowanie układu FPGA umożliwia wykonywanie operacje w sposób współbieżny. Należy pamiętać, iż użycie procesora pozwoliłoby w tym przypadku na wykonanie co najwyżej jednego porównania na takt zegara. Układ FPGA daje możliwość dokonania nawet kilkunastu tysięcy równoczesnych porównań. Zmniejsza to czas obliczeń o te kilkanaście tysięcy razy. Nie bez znaczenia jest również wielkość przetwarzanych liczb. Procesor 32-bitowy jest wyposażony w zbiór instrukcji dokonujących szybkich obliczeń na liczbach 32-bitowych, natomiast przy liczbach nawet o jeden bit większych obliczenia stają się dużo bardziej złożone i mniej efektywne. Do liczb większych niż 32 bity można oczywiście użyć procesora 64-bitowego, natomiast w tym przypadku zwiększenie rozmiaru haszu powyżej 64-bitów również znacząco komplikuje obliczenia. Zastosowanie w tym przypadku układu FPGA jest bardzo korzystne, gdyż umożliwia przetwarzania liczb o dowolnej długości bitowej, bez znaczącego wpływu na częstotliwość graniczną.

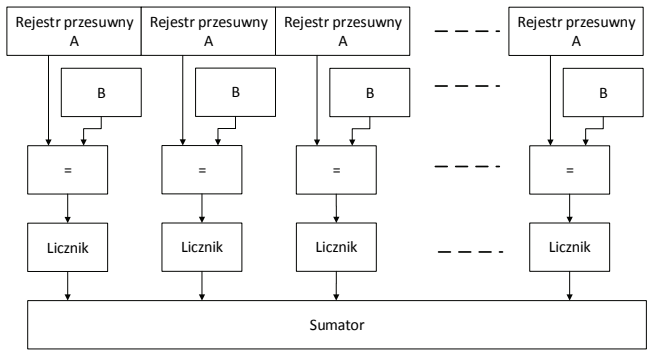
3. Propozycje realizacji sprzętowej metryk

Wszystkie zaimplementowane algorytmy opierają się w schemacie blokowym z rys. 1.



Rys. 1. Ogólny schemat blokowy metryki
Fig. 1. General block diagram of the metric

Zaprezentowany algorytm porównuje dwa zbiory liczb metodą „każdy z każdym". Zbiory te nazwano *A* i *B*. Liczby ze zbioru *A* stanowią odcisk palca dokumentu, który podlega procesowi klasyfikacji. Są one sekwencyjnie podawane na wejście rejestru przesuwnego, co takt zegara. Liczby zbioru *B* stanowią zbiór haszy wygenerowany z dużej liczby dokumentów o jednakowej tematyce. Są to hasze wzorcowe, na podstawie których dokument będzie klasyfikowany. Ponadto w schemacie blokowym można wyróżnić szereg komparatorów, których głównym zadaniem jest porównanie ze sobą liczby ze zbioru *A* z liczbą ze zbioru *B*. Są one oznaczone symbolem „=". Komparator, wraz z rejestrami zawierającymi liczby *A* i *B* stanowią blok porównujący. Liczba takich bloków występuje w kodzie źródłowym jako parametr i może zostać dowolnie zwiększona. Jedynym ograniczeniem jest w tym przypadku ilość dostępnych zasobów wybranego układu FPGA. Każdy z takich bloków dokonuje porównań w sposób współbieżny z pozostałymi. Liczba pozytywnych porównań jest zliczana w bloku „Adder". Zaproponowane modele metryk zostały zaprogramowane w języku VHDL. Założeniem projektowym było uzyskanie jak największej częstotliwości maksymalnej, nawet kosztem zajętości zasobów. Powstały dwie alternatywne architektury. W rozwiązaniu z rys. 2, na wyjściu każdego komparatora znajduje się licznik. Licznik ten jest inkrementowany, gdy na wejściu podległego mu komparatora pojawią się dwie takie same wartości. Aby uzyskać liczbę pozytywnych porównań, należy, po zakończonym procesie przetwarzania, zsumować wartości wszystkich liczników.



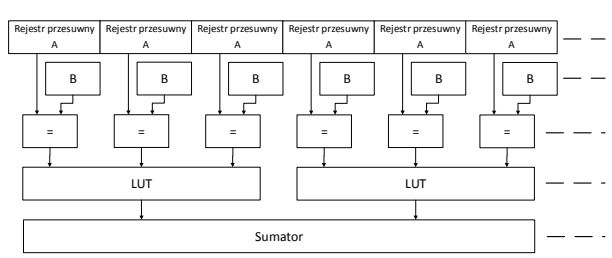
Rys. 2. Schemat blokowy metryki z wykorzystaniem liczników
Fig. 2. Block diagram of the metric with counters

Drugi model metryki (rys. 3) zakłada wykorzystanie tablicy LUT (ang. lookup table). Jeden blok LUT jest podłączony do wyjść kilku komparatorów. Na wyjściu każdego komparatora może pojawić się jeden bit danych. Jeżeli obie liczby wejściowe są takie same, komparator generuje „1", natomiast w przeciwnym przypadku „0". Na wejściu bloku LUT pojawia się więc pewna sekwencja zer i jedynek, która jest zależna od aktualnego stanu komparatorów. Na wyjściu bloku jest generowana bezpośrednio liczba jedynek w sekwencji wejściowej. Na przykład założymy, że blok LUT posiada cztery wejścia, oraz że na trzech z nich pojawi się sygnał „1". Przykładowe permutacje na wejściu mogą wyglądać następująco: „0111", „1101", „1011" itd. Dla każdej powyższej sekwencji na wyjściu bloku pojawi się liczba $3_{10} = 11_2$. Przykładowa tablica LUT dla wejścia dwubitowego została przedstawiona w tab. 1.

Tab. 1. Przykładowa tablica LUT dla wejścia dwubitowego
Tab. 1. Example of LUT for a 2-bit input

Liczba binarnie	Liczba dziesiętnie	Ilość jedynek
00	0	0
01	1	1
10	2	1
11	3	$2_{10} = 10_2$

Aby lepiej wykorzystać dostępne zasoby sprzętowe, w projekcie przyjęto długość wektora wejściowego równą 6. Długość wektora wyjściowego wynosi 3 bity, co znacząco upraszcza układ sumatora. Aby wyznaczyć liczbę pozytywnych porównań należy w każdym takcie zegara sumować liczby z wyjść bloków LUT.

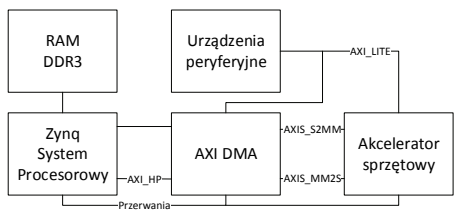


Rys. 3. Schemat blokowy metryki z wykorzystaniem LUT
Fig. 3. Block diagram of the metric with LUT

4. Projekt akceleratora

Schemat blokowy systemu mikroprocesorowego z podłączonym akceleratorem został zaprezentowany na rys. 4. Całym układem zarządza system procesorowy platformy Zynq. Składają się na niego dwa rdzenie ARM Cortex-A9 [4]. System procesorowy komunikuje się z peryferiami za pomocą dwóch magistrali AXI.

Magistrala AXI-Lite służy do konfiguracji układów peryferyjnych. Magistrala AXI-HP (High Performance) jest podłączona do kontrolera pamięci DDR3 RAM, udostępniając bezpośredni dostęp do tej pamięci z pominięciem procesora. Dane z tej pamięci są pobierane przez kontroler AXI DMA, a następnie zamieniane na strumień danych, z wykorzystaniem magistrali AXI-Stream. Dane z DMA są następnie pobierane przez blok akceleratora sprzętowego, gdzie dokonywane są wymagane obliczenia. Wyniki tych obliczeń są przesyłane w postaci strumieniowej do bloku DMA, którego zadaniem jest ich zapisanie w wybranych komórkach pamięci RAM systemu mikroprocesorowego. Tak zaprojektowany układ akceleratora daje możliwość maksymalnego odciążenia procesora. Aby rozpocząć proces akceleracji należy w pierwszej kolejności zapisać blok danych wejściowych w odpowiednim bloku pamięci RAM. Kolejnym krokiem jest konfiguracja bloku akceleratora za pomocą magistrali AXI-LITE. Jeżeli są wykorzystywane przerwania od akceleratora i kontrolera DMA, to należy je włączyć. Następnie należy skonfigurować blok DMA, określając między innymi adres początku bloku danych przeznaczonych do wysyłki, oraz ich ilość, a następnie adres, pod którym ma zostać zapisany wynik akceleracji, oraz ilość danych, która ma zostać odebrana. Po wykonaniu powyższych operacji rozpocznie się proces przetwarzania danych, a jego zakończenie zostanie zasygnalizowane odpowiednim przerwaniem. Jak łatwo zauważyć, cały proces wykonuje się z całkowitym pominięciem procesora, który w tym czasie może zająć się na przykład przygotowaniem danych przetwarzanych w następnej kolejności. Praktyczny układ został zrealizowany na płytce prototypowej ZedBoard. Projekt systemu powstał z wykorzystaniem narzędzia Xilinx Platform Studio (XPS). Podczas budowy systemu zaprojektowano własny uniwersalny blok IP (IP Core), za pomocą którego można w łatwy sposób dołączyć dowolny akcelerator sprzętowy. Blok ten został wyposażony w dwie magistrale typu AXI-Stream (AXIS_S2MM i AXIS_MM2S), pracujące odpowiednio w trybie master dla danych wejściowych i w trybie slave dla danych wyjściowych, oraz magistralę AXI-Lite, który może zostać wykorzystany w dowolny sposób, np. do konfiguracji akceleratora.



Rys. 4. Schemat blokowy akceleratora sprzętowego
Fig. 4. Block diagram of the hardware accelerator

Podczas budowania systemu zaimplementowano przykładowy model akceleratora na bazie sumatora. W przyszłości zostanie on zastąpiony modelem obliczającym metrykę podobieństwa tekstu. W celu uproszczenia obsługi akceleratora została zaprojektowana biblioteka w języku C. Udostępnia ona funkcje wspomagające proces akceleracji (konfiguracja akceleratora, włączanie i wyłączanie akceleracji, obsługa przerwań itp.). Na bazie biblioteki powstały dwie wersje oprogramowania. Pierwsza wersja powstała bez systemu operacyjnego. Program kopiuje blok danych pod wskazany adres, włącza akcelerację a następnie odczytuje jej wynik z innego miejsca w pamięci. Nad poprawnością transmisji czuwa system przerwań systemu procesorowego. Program został dokładnie przetestowany i spełnia swoje założenia. Druga wersja oprogramowania powstała jako aplikacja dla systemu operacyjnego Petalinux. Zmieniony został w niej sposób dostarczania danych do akceleratora. Założono tutaj, że akcelerator będzie operował na plikach. Takie podejście ma wiele zastosowań praktycznych. Można na przykład wykorzystać sprzętową akcelerację do szyfrowania plików. Oprogramowanie wczytuje zawartość pliku do pamięci. Następnie włączany jest proces akceleracji. Jej wynik jest pobierany z pamięci i zapisywany do pliku wyjściowego.

5. Wyniki implementacji

Wyniki implementacji metryki (wersja z licznikami):
Liczba bloków porównujących: 1000
Implementacja dla układu XC7VX485T
Częstotliwość graniczna: 87MHz

Tab. 2. Wyniki implementacji pierwszej wersji metryki
Tab. 2. Implementation results for the first version of the metric

Rodzaj zasobu	Użyte	Dostępne	Procent zajęcia
Slice Registers	43094	607200	7%
Slice LUTs	63159	303600	20%
LUT-FF	33570	72683	46%

Wyniki implementacji metryki (wersja z LUT):
Liczba bloków porównujących: 1000
Implementacja dla układu XC7VX485T
Częstotliwość graniczna: 114MHz

Tab. 3. Wyniki implementacji drugiej wersji metryki
Tab. 3. Implementation results for the second version of the metric

Rodzaj zasobu	Użyte	Dostępne	Procent zajęcia
Slice Registers	36428	607200	5%
Slice LUTs	44329	303600	14%
LUT-FF	33610	47147	71%

Wyniki implementacji akceleratora:
Implementacja dla układu XC7Z020CLG484.
Częstotliwość graniczna: 118.259MHz

Tab. 4. Wyniki implementacji akceleratora sprzętowego
Tab. 4. Implementation results for the hardware accelerator

Rodzaj zasobu	Użyte	Dostępne	Procent zajęcia
Slice Registers	3315	106400	3%
Slice LUTs	2942	53200	5%
Użyte jako logika	2463	53200	4%
Użyte jako pamięć	210	17400	1%
Liczba użytych Slice'ów	1127	13300	8%
RAMB36E1/FIFO36E1s	3	140	2%
RAMB18E1/FIFO18E1s	1	280	1%

6. Podsumowanie i przyszłe badania

Opisywany projekt jest w fazie ciągłego rozwoju. Do tej pory zaimplementowano dwa działające sprzętowe modele do obliczania metryk podobieństwa tekstów. Oba modele zostały dokładnie przesymulowane i działają zgodnie z założeniami. W czasie trwania projektu powstał również uniwersalny układ akceleratora sprzętowego, który może zostać łatwo zintegrowany z dowolnym sprzętowym układem akceleracji. Podczas dalszych prac planowane zintegrowanie tego systemu z blokiem do obliczania metryk podobieństwa tekstów.

Projekt został sfinansowany ze środków Narodowego Centrum Nauki przyznanych na podstawie decyzji numer DEC-2011/01/B/ST6/03024.

7. Literatura

[1] Wielgosz M., Koryciak S., Janiszewski M., Pietroń M., Russek P., Jamro E., Dąbrowska-Boruch A., Wiatr K.: Parallel MPI implementation of N-gram algorithm for document comparison. ACACES 2013 : the 9th international summer school on Advanced Computer Architecture and Compilation for High-performance and Embedded Systems, Ghent: Academia Press, pp.217-220, 2013.
[2] Parapar J., Barreiro A.: Evaluation of Text Clustering Algorithms with N-Gram-Based Document Fingerprints. IRLab, Computer Science Department University of A Coruna, Spain.
[3] Norzima Elbegbayan: Winnowing, a Document Fingerprinting Algorithm, TDDC03 Projects, Spring 2005.
[4] http://www.xilinx.com

otrzymano / received: 12.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*