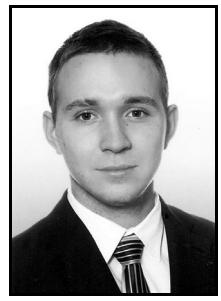
Maciej BARSZCZOWSKI, Sebastian KORYCIAK, Agnieszka DĄBROWSKA-BORUCH, Kazimierz WIATR
AGH AKADEMIA GÓRNICZO-HUTNICZA W KRAKOWIE, ACK CYFRONET AGH, ul. Nawojki 11, 30-950 Kraków
AGH AKADEMIA GÓRNICZO-HUTNICZA W KRAKOWIE, KATEDRA ELEKTRONIKI, Al. Mickiewicza 30, 30-059 Kraków

Komunikacja ze sprzętowym akceleratorem haszowania n-gramów dla procesora ARM z wykorzystaniem portu ACP

Inż. Maciej BARSZCZOWSKI



Ukończył studia pierwszego stopnia na AGH (2013), wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej na kierunku Inżynieria Biomedyczna. Obecnie kończy studia magisterskie o specjalności Informatyka i Elektronika Medyczna. Jego zainteresowania to implementacja algorytmów sterowania sygnałami biologicznymi, budowa interfejsów człowiek – maszyna przy pomocy układów programowalnych.

e-mail: barszcz2@gmail.com

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

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

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 uruchomienie portu ACP w układzie EPP firmy Xilinx przy użyciu CDMA zarządzającego transmisją pomiędzy akceleratorem, a rdzeniami procesora. Głównym celem badań było utworzenie modułu dokonującego tak zwanego haszowania zbiorów danych. Do wykonania tej operacji wykorzystany został układ Zynq 7000 posiadający zasoby logiki programowalnej oraz dwa rdzenie ARM A9. Powstały dwie koncepcje realizacji akceleratora. Pierwsza wersja zakładała bezpośredni przepływ danych ze źródła do akceleratora, a następnie do rdzeni ARM. Drugie rozwiązanie zakłada wykorzystanie portu ACP.

Słowa kluczowe: ACP, akceleracja sprzętowa, ARM, Zynq.

Communication with an n-gram hashing hardware accelerator for the ARM using ACP

Abstract

This paper introduces a new approach to hardware acceleration using the ACP(Acceleration Coherency Port) in Xilinx Zynq-7000 EPP XC7Z020. The first prototype allocated BRAM memory and transferred data through the ACP. The second one used a hardware hashing module to process data outside the CPU. The module received and returned data through the ACP port. The main task of the system is to replace a set of data with its shorter representative of constant length without interference of the processing unit. The main benefit of hashing data lies within the constant length of function outcome, which leads to data compression. Compression is highly desirable while comparing large subsets of data, especially in data mining. The execution of a hashing function requires high performance of the CPU due to the computational complexity of the algorithm. Two concepts where established. The first one assumed transferring data directly do the hardware accelerator and later to ARM cores. This solution is attractive due to its simplicity and relatively fast. Unfortunately, the data cannot be processed before hashing with the same CPU without significant speed reduction. The second approach used the ACP port which can transfer data very fast between L2/L3 cache memory without flushing of validating cache. The data can be processed by the software driven CPU, sent to the accelerator and then sent back to CPU for further processing. To accomplish the established task, the Zynq 7000 EPP with double ARM A9 core and programmable logic in one chip was used.

Keywords: ACP, acceleration, ARM, Zynq.

1. Wstęp

Dane to podstawowa dewiza naszej epoki, informacja jest w dzisiejszych czasach bardzo cenna a pozyskanie informacji w dobie rozwoju ogromnej ilości systemów jej przedstawiania stanowi niebagatelną kwestię. Ogromne ilości treści są przesyłane oraz analizowane. Dla uzyskania optymalnych wyników transferu danych oraz ich analizy, klasyfikacji, porównywania wykorzystuje się metody kompresji takie jak funkcja skrótu. Zbiory danych mogą zostać zastąpione przez ich skróty – przedstawicieli o stałym, pomniejszonym rozmiarze. Funkcje haszujące dane implementowane są głównie w oprogramowaniu co ze względu na ich charakter wymaga poświecenia dużej ilości zasobów. Innym rozwiązaniem jest implementacja funkcji skrótu bezpośrednio w sprzęcie. Procesor może zostać zwolniony z obowiązku haszowania danych i wykonywać inne instrukcje równolegle z akceleratorem [7]. Wykorzystanie najnowszych rozwiązań hybrydowych łączących w sobie klasyczne rdzenie ARM oraz zasoby sprzętowe FPGA pozwala na implementacje złożonego systemu w jednym układzie. ACP (Acceleration Coherency Port) pozwala na bardzo szybkie przekazywanie danych z procesora do sprzętu i z powrotem praktycznie bezpośrednio z pamięci podręcznej rdzeni. Kompresja wykonywana przez moduł sprzętowy (akcelerator) redukuje ilość nadmiarowych danych tworząc reprezentację danych łatwą do przechowania i porównania ze wzorcem. Cecha ta może okazać się przydatna przy budowie systemów analizy treści oraz eksploracji danych [1].

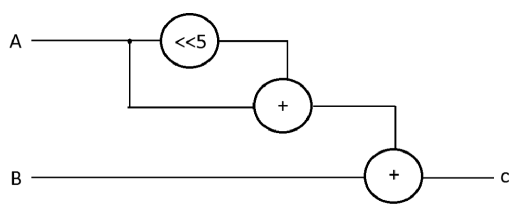
2. Układ Zynq7000

Do osiągnięcia założonych celów wykorzystano układ Zynq 7000 XC7Z020 [4] posiadający zasoby logiki programowalnej oraz dwa rdzenie ARM A9. Układ ten znajduje się w zestawie uruchomieniowym ZedBoard [3]. Poza wykonywaniem funkcji skrótu może on zostać wykorzystany do innych zadań związanych

z przetwarzaniem danych lub zarządzaniem projektami. Celem budowy akceleratora jest zwolnienie zasobów procesora, które w klasycznym podejściu były by silnie wykorzystane przez funkcje skrótu. Jako że haszowanie składa się z prostych operacji matematycznych powielanych wielokrotnie dla danego zbioru danych idealną alternatywą dla podejścia klasycznego jest wykorzystanie logiki programowalnej. Najprostsza realizacja zakłada przesłanie danych bezpośrednio do akceleratora ze źródła zewnętrznego, co może zostać zrealizowane w prosty sposób dzięki zasobom gotowych modułów przygotowanych dla układów Xilinx. Jakkolwiek rozwiązanie to jest proste w implementacji oraz wykazuje znacznie większa efektywność gdyż dane zostają przesłane tyko raz na drodze PL – CPU to uniemożliwia ono wcześniejsze przetworzenie danych wejściowych pod kontrolą programową. Wykorzystane w układzie Zynq 7000 rozwiązania umożliwiły przesłanie danych bezpośrednio z jednostki Snoop nadzorującej prace pamięci Cache do sprzętowego akceleratora przy użyciu portu ACP(Acceleration Coherency Port) – 64 bitowej magistrali AXI pracującej w trybie koherentnym z pamięcią Cache procesora. Możliwe jest uprzednie utworzenie n-gramów na przykład metodą okna SWA-03, która wybiera stosowny wzorzec do porównywania zbiorów danych. Wykorzystanie metod opartych o operacje na n-gramach wykonane jest pod kontrolą programową przez rdzenie ARM i stanowi wstępną obróbkę danych przed ich przekazaniem do akceleratora. Do dyspozycji pozostały liczne zasoby peryferii związanych z rdzeniami ARM.

3. Funkcja skrótu

Wykorzystana do kompresji danych funkcja skrótu (rys. 1) to prosty algorytm zakładający wykonanie serii operacji matematycznych. Porcje danych podlegają przesunięciu oraz sumie. W ten sposób dokonuje się kombinacji czterech fragmentów porcji danych, gdzie pierwszy fragment ulega kombinacji ze stałym kluczem. Schemat ideowy funkcji skrótu przedstawia rysunek 1. W bardziej zaawansowanych rozwiązaniach związanych z operacjami na zbiorach danych wykorzystuje się kodowanie Huffmana lub filtr Blooma. Sprzętowe implementacje tych rozwiązań rozwijane w ramach projektu Synat ACK Cyfronet AGH mogą zastąpić funkcje skrótu [2].



Rys. 1. Idea działania funkcji haszującej
Fig. 1. The main idea of the hash function

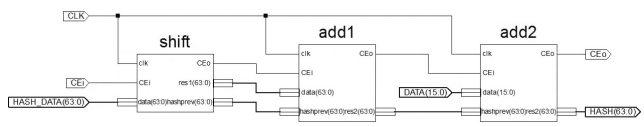
Pierwsza implementacja modułu sprzętowego charakteryzowała się prostym modelem behawioralnym i sporym wykorzystaniem zasobów logiki programowalnej. Nowy strukturalny model umieszczony na rys. 3 oszczędza zasoby oraz pozwala na zwiększenie szybkości działania o 20% względem poprzedniego modelu. Każda operacja zdefiniowana jest jako osobny moduł, którego schemat modułowy przedstawia rys. 2. W bloku MAIN wykorzystuje się 4 moduły HASH.

4. Wyniki implementacji

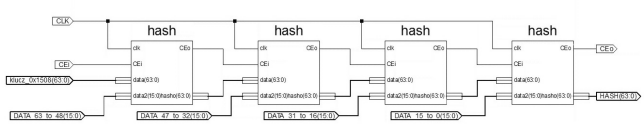
Wszystkie moduły zostały napisane w języku opisu sprzętu VHDL. W tab. 1 zostały zaprezentowane wyniki syntezy obu modeli. Częstotliwości możliwe do osiągnięcia wskazują na to, że ograniczenie prędkości przepływu danych będzie stanowić taktowanie magistrali AXI. Zgodnie z przewidywaniami uzyskano wzrost szybkości działania modelu strukturalnego względem modelu behawioralnego. Implementacja wykazała także znaczny wzrost wykorzystania rejestrów oraz zmniejszenie ilości tablic LUT. Wybrano model strukturalny, którego podstawowymi jednostkami budulcowymi są moduły shift oraz add realizujące proste operacje na zbiorach danych. Połączenie modułów shift oraz add w ramach bloku wykonującego funkcję skrótu na bloku danych prezentuje rysunek 2. Blok funkcji skrótu nazwany modułem hash wykorzystany jest czterokrotnie – za każdym razem dla kolejnej porcji danych wydzielonych ze zbioru otrzymanego na wejściu. Rysunek 3 przedstawia połączenie modułów hash w akceleratorze.

Tab. 1. Wyniki implementacji modułów dla układu xc7z020-2clg484
Tab. 1. Module implementation results for xc7z020-2clg484

	Model behawioralny	Model strukturalny
Tablice LUT	255	199
Rejestry	30	291
Częstotliwość	554 MHz	664 MHz



Rys. 2. Schemat blokowy i sposób połączenia sygnałów w modułu HASH
Fig. 2. Block diagram and connection scheme in the HASH module



Rys. 3. Schemat blokowy i sposób połączenia sygnałów w modułu MAIN
Fig. 3. Block diagram and connection scheme in the MAIN module

5. Uruchomienie portu ACP

Port ACP podobnie jak znaczna część zasobów powiązanych z układami ARM jest widoczny z systemu poprzez adresy fizyczne [7]. Jakkolwiek adresowanie zasobów adresem fizycznym było by zadaniem prostym użytkownik nie posiada takiej możliwości wykorzystując narzędzia programowe. Z uwagi na rozpiętość wirtualnej przestrzeni adresowej wykorzystywany jest moduł MMU (Memory Management Unit). MMU zarządza przestrzenią adresową i umożliwia rzutowanie adresów wirtualnych na fizyczną przestrzeń adresową a dokładniej dzielenie adresów logicznych na strony oraz ich tłumaczenie na adresy fizyczne przy pomocy TLB (Translation Lookaside Buffer). Przed podjęciem pracy z ACP należy skonfigurować TLB tak aby źródło danych było osiągalne dla koherentnego transferu przez ACP. Istotnym parametrem jest konfiguracja związana z pamięcią cache. Podczas transferu danych należy wykorzystywać przestrzeń adresową leżącą w przestrzeni pamięci portu ACP. Pierwsze próby uruchomienia portu ACP oparte były na prostym systemie wymiany danych pomiędzy dwoma zbiornikami. Ze strony systemu związanego z rdzeniami ARM zbiornik danych stanowiła pamięć OCM (On Chip Memory). Moduł OCM zawarty w układzie Zynq zawiera blok pamięci RAM o pojemności 256 kilobajtów. Pamięć OCM posiada 2 porty AXI, z czego jeden jest portem dedykowanym do operacji wykonywanych za pośrednictwem jednostki Snoop Control dzięki czemu dostęp do danych poprzez procesor jest

szybki i nie zakłócony przez inne moduły układu Zynq. Drugim zbiornikiem danych była pamięć BRAM utworzona w zasobach logiki programowalnej. Transfer danych sterowany przez moduł CDMA (Central Direct Memory Access) umożliwiał łatwą programową kontrolę operacji. CDMA – gotowy moduł zapewniony razem ze środowiskiem Xilinx Platform Studio przesyła dane pomiędzy dowolnymi instancjami bez ingerencji procesora w dane. Transfer wykonuje się w 4 operacjach, najpierw następuje wysłanie bajtu konfiguracyjnego, następnie bajtu z adresem źródłowym. Kolejny bajt to adres przeznaczenia, natomiast ostatni to ilość danych jaka ma zostać przeniesiona. Dostęp do pamięci OCM przez port ACP wymaga odpowiedniej konfiguracji jednostki MMU (Memory Management Unit) do mapowania adresu wirtualnego na adres fizyczny znajdujący się w zakresie portu ACP. Kolejnym etapem było podłączenie wcześniej zaprojektowanego akceleratora do modułu OCM poprzez port ACP. Podobnie jak poprzednio do transferu danych wykorzystano CDMA. W oparciu o uproszczone magistrale AXI Lite połączono w całość elementy prototypu. Testy wypadły pomyślnie i uzyskano skrót danych testowych. Dane zostały pobrane z pamięci OCM, następnie po przetworzeniu zostały do niej odesłane. Operacje te sterowane były programowo przy pomocy CDMA. Najnowsza wersja zakłada wykorzystanie usprawnionej magistrali o poszerzonej przepustowości dla uzyskania maksymalnej szybkości działania.

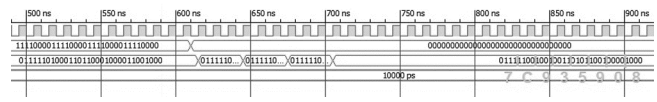
6. Symulacja i wyniki działania

Przytoczony poniżej kod prezentuje zrzut ekranu z konsoli programu Xilinx SDK podczas wykonywania programu testowego zapisującego w bloku OCM3 cztery bajty o wartości 0xF0. Po wykonaniu transferu danych w 2 kierunkach przez CDMA wartość OCM odczytywana jest ponownie.

```
Hello World
... ocm memory before write
0xFFFF800 = 0xF0
0xFFFF801 = 0xF0
0xFFFF802 = 0xF0
0xFFFF803 = 0xF0
... DMA config
...STATUS...
0x40000004 = 0x00001000
...STATUS...
0x40000004 = 0x00001002
... DMA config
...STATUS...
0x40000004 = 0x00001000
...STATUS...
0x40000004 = 0x00001002
... ocm memory after read
0xFFFF800 = 0x08
0xFFFF801 = 0x59
0xFFFF802 = 0x93
0xFFFF803 = 0x7C
```

Rys. 4. Zrzut ekranu z konsoli Xilinx SDK
Fig. 4. Xilinx SDK console screenshot

Rys. 5 to zrzut ekranu z przeprowadzonej symulacji, mającej na celu odwzorowanie realnego działania akceleratora pozbawionego sygnałów kontrolnych. Przez pewną ilość czasu na wejście akceleratora podawane są dane. Po upływie określonego czasu na wejście podawane są dane zerowe. W ciągu 100ns od zmiany stanu wejścia, na wyjściu akceleratora pojawiają się cztery bajty będące wynikiem akceleracji danych zerowych. Są to dokładnie te same dane które uzyskaliśmy w wyniku działania realnego projektu sprzętowego. Ze względu na niedoskonałości zaprojektowanego systemu w dalszym etapie należy zwrócić uwagę na integracje sygnałów kontrolnych akceleratora z magistralą lub ich inne doprowadzenie do jednostki PS7. Dla zwiększenia wydajności magistrala AXI4 Lite powinna zostać zastąpiona pełnym portem AXI a szerokość szyny danych zwiększona do 64 bitów. Aby uniknąć spadku wydajności związanego z brakiem kompatybilności operacji pomiędzy pamięcią OCM a portem ACP należy użyć bezpośrednio pamięci cache modyfikując zapis tablicy L1 w module MMU. Po optymalizacji sprzętowej kolejnym krokiem będzie integracja projektu sprzętowego z systemem operacyjnym poprzez aplikację do obsługi sprzętu pod kontrolą systemu Linux.



Rys. 5. Przebiegi z symulacji modułu MAIN
Fig. 5. Waveforms from simulation of the MAIN module

7. Wnioski

W ramach opracowanego rozwiązania osiągnięto założone cele i przebadano nową koncepcje analizy danych. Wykorzystanie portu ACP przynosi wymierne korzyści związane z przyśpieszeniem obliczeń. Uniwersalność portu i jego konstrukcja umożliwia wykorzystanie go z praktycznie dowolnym modułem sprzętowym, którego optymalna konstrukcja umożliwia zwiększenie szybkości i rzetelności działania w stosunku do innych modeli. Daje to niebywałą okazję to rozwoju technik do tej pory wymagających dużych zasobów czasu oraz sprzętu. W połączeniu z rozwiązaniami sprzętowymi opracowanymi w ramach projektu Synat przez ACK Cyfronet AGH sprzętowa akceleracja z wykorzystaniem portu ACP stwarza nadzieje na nowe osiągnięcia w dziedzinie przetwarzania danych.

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

8. Literatura

[1] Jamro E., Wielgosz M., Russek P., Pietroń M., Żurek D., Janiszewski M., Wiatr K.: Implementation of algorithms for fast text search and files comparison. Proceedings the High Performance Computer Users Conference KU KDM 2013, Academic Computer Centre Cyfronet AGH, Kraków, pp. 83-84, 2013.
[2] 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.
[3] http://zedboard.org/documentation/1521
[4] http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf
[5] Bryan Mealy, Fabrizio Tappero: Free Range VHDL. The no-frills guide to writing powerful code for your digital implementations, 1 April 2013, freerangefactory.org
[6] Bryan J. Mealy & James T. Mealy: Digital McLogic Design, Bryan J. Mealy 2012.
[7] Mohammadsadegh Sadri, Christian Weis, Norbert Wehn, and Luca Benini, Energy and performance exploration of accelerator coherency port using Xilinx ZYNQ. In Proceedings of the 10th FPGAworld Conference, 2013.

otrzymano / received: 22.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*