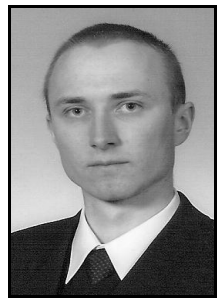
Maciej WIELGOSZ, Ernest JAMRO, Kazimierz WIATR
AKADEMIA GÓRNICZO-HUTNICZA, ACK - CYFRONET
AKADEMIA GÓRNICZO-HUTNICZA, KATEDRA ELEKTRONIKI

Implementacja w układach FPGA operacji eksponenty dla liczb w standardzie IEEE-754 o podwójnej precyzji

Mgr inż. Maciej WIELGOSZ



Ukończył studia na AGH (2005), na wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obecnie jest doktorantem w Katedrze Elektroniki AGH i bierze czynny udział w pracach badawczych realizowanych w zespole rekonfigurowalnych systemów obliczeniowych. Jego zainteresowania naukowe dotyczą sprzętowej akceleracji obliczeń, kompresji obrazu i sieci neuronowych.

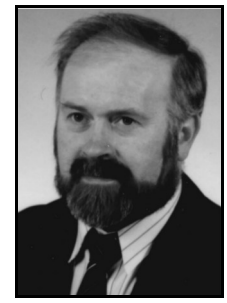
e-mail: wielgosz@agh.edu.pl

Dr inż. Ernest JAMRO

Ukończył studia na AGH na kierunku Elektronika oraz na University of Huddersfield (UK) na kierunku Elektronika i Telekomunikacja. Obronił pracę doktorską w 2001 roku na AGH na wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki. Aktualnie jest adiunktem w Katedrze Elektroniki na AGH. Jego zainteresowania naukowe to sprzętowa akceleracja obliczeń, niskopoziomowe przetwarzanie obrazów, sieci neuronowe.

e-mail: jamro@agh.edu.pl

Prof. dr hab. inż. Kazimierz WIATR



Studia AGH Kraków (1980), dr nauk technicznych (1987), dr habilitowany (1999) i profesor (2002). Profesor zwyczajny na Akademii Górniczo-Hutniczej oraz Dyrektor Akademickiego Centrum Komputerowego 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

W artykule przedstawiono implementację operacji obliczania eksponenty o podwójnej precyzji obliczeń w układach FPGA. Zaproponowano metodę tablicowo – aproksymacyjną, dla której wykorzystano 3 niezależne tablice 512×64-bity do obliczenia 27 najstarszych bitów mantysy oraz aproksymacje wielomianową ex≈1+x dla pozostałych bitów mantysy. Wyniki implementacji pokazują że proponowany moduł zajmuje około 7.5% układu Virtex-4 LX200.

Słowa kluczowe: obliczanie funkcji elementarnych, przyspieszanie obliczeń, układy programowalne.

FPGA Implementation of Exponent Function for Double Precision IEEE-754 Standard

Abstract

This paper presents FPGA implementation of exponent operation in double precision format. A mixture of Look-Up Table (LUT) and approximation methods was employed. Twenty seven most significant bits of input mantissa are calculated employing 3 independent LUTs, the rest input bits are calculated by approximation: ex≈1+x. Implementation results in roughly 7.5% occupation of Virtex-4 LX-200.

Keywords: elementary functions computations, computing acceleration, programmable devices.

1. Wstęp

Numeryczna analiza skomplikowanych modeli fizycznych oraz chemicznych realizowana np. w pakiecie Gaussian wymaga dużej mocy obliczeniowej. Na szczególną uwagę zasługuje tutaj operacja eksponenty, która jest powszechnie wykonywana natomiast nie jest bezpośrednio wspierana przez procesory ogólnego przeznaczania. W tym miejscu należy podkreślić, że wykonanie pojedynczej operacji eksponenty wymaga w takich systemach wiele taktów zegara systemowego. Dlatego kluczowym celem, jaki postawili sobie autorzy niniejszej pracy było wyraźne przyspieszenie realizacji operacji eksponenty dzięki dedykowanej a przez to wydajnej architekturze sprzętowej realizowanej w układach programowalnych FPGA.

Moduł sprzętowy oparty na układzie FPGA będzie ściśle współpracował z częścią programową większego systemu obliczeniowego. Metoda realizacji obliczeń w prezentowanym systemie opiera się na płynnej wymianie danych pomiędzy częścią programową oraz sprzętową, jest to rozwiązanie typowe w dziedzinie HPC.

Rekonfugurowalne układy FPGA, odpowiednio zaprogramowane umożliwiają wykonywanie wybranych operacji dużo szybciej i wydajniej niż procesory ogólnego przeznaczenia. Dotyczy to szczególnie operacji stałoprzecinkowych o ograniczonej precyzji wykonywanych na dużej liczbie danych, np. podczas filtracji obrazu [1]. Niestety wykonywanie operacji zmiennoprzecinkowych o podwójnej precyzji wymaga znacznych zasobów układu FPGA i dlatego do niedawna układy FPGA nie były wykorzystywane do wspomagania obliczeń numerycznych pełnej precyzji. Rozważano natomiast wykonywanie operacji zmiennoprzecinkowych o mniejszej precyzji. Niestety kryterium odgrywającym ogromną rolę we współczesnych systemach HPC jest dokładność obliczeń i duża precyzja danych tworzących modele obliczeniowe. Kierując się wyżej wspomnianymi potrzebami w ramach pracy zaprojektowano moduł obliczeniowy realizujący operacje exp() w standardzie IEEE-754 o podwójnej precyzji.

2. Funkcja eksponenty

Techniki obliczania elementarnych funkcji mogą zostać zakwalifikowane do dwóch grup: iteracyjne oraz nieiteracyjne. Do implementacji została wybrana metoda nieiteracyjna, która może być realizowana następującymi sposobami:
- metody tablicowe - Look-Up Table (LUT),
- metody aproksymacyjne,
- metody mieszane (tablicowo-aproksymacyjne).

Metody tablicowe są najprostsze koncepcyjnie i zarazem wyniki ich implementacji skutkują bardzo szybką pracą układu wynikowego. Zasadniczą wadą tego rozwiązania jest jednak spora ilość zasobów zajmowanych przez tablice LUT przechowujące wartości funkcji, która wzrasta wykładniczo wraz z precyzją danej wejściowej. W konsekwencji wielkość pamięci staje się niedopuszczalna w przypadku danej wejściowej szerszej niż około 16-bitów.

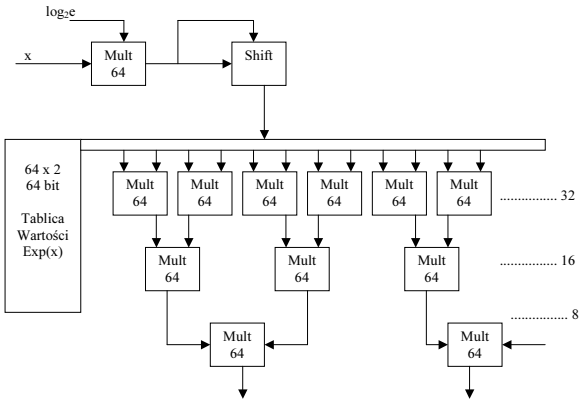
Istnieje sporo implementacji metod tablicowych obliczania funkcji exp(), lecz projekt zrealizowany w Ames Laboratory [2] zwraca szczególną uwagę ze względu na zbliżony profil prac badawczych oraz stosowaną 64-bitową precyzje obliczeń. We wspomnianej pracy zostały wykorzystane bardzo dobrze znane zależności:

$$e^x = 2^{x*log2e} \tag{1}$$

oraz

$$e^{x1+x2} = e^{x1} \cdot e^{x2}. \tag{2}$$

Zastosowanie podstawy liczby *2* zamiast *e* zdecydowanie upraszcza obliczenie wartości 2^x dla x_c całkowitych, ponieważ wartość x_c bezpośrednio przekłada się na wartość eksponenty wyniku. W konsekwencji wartość x jest dzielona na część całkowitą x_c i ułamkową x_u. Część całkowita x_c jest zapisywana bezpośrednio do eksponenty wyniku. Część ułamkowa x_c jest natomiast wykorzystywana dalej do obliczenia mantysy wyniku. Schemat wewnętrzny logiki [2] został przedstawiony na rys. 1. Można zaobserwować jak dużą część zajmowanych zasobów stanowią układy mnożące zmiennoprzecinkowe.

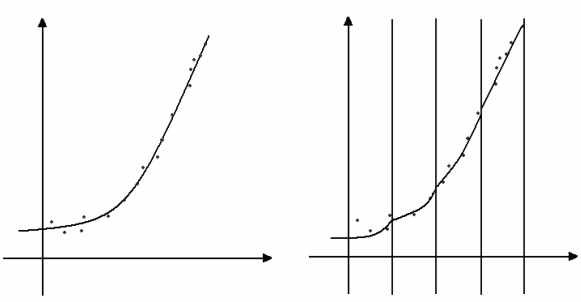


Rys. 1. Tablicowa implementacja modułu obliczania funkcji exp()
Fig. 1. Table-based method implementation of exp() function

Powyższe rozwiązanie wymaga implementacji 64 mnożarek zmiennoprzecinkowych. Mnożenia te są realizowane w architekturze sześciostopniowego drzewa binarnego. W zależności od sposobu realizacji układów mnożących, układ osiąga teoretyczne opóźnienie potokowe (latency) rzędu 57-162 taktów zegara. Powyższa implementacja absorbuje prawie wszystkie zasoby jednego z większych układów: Virtex-II Pro (55,616 slice'ów).

Powszechnie stosowanymi metodami są również aproksymacje wielomianowe [3]. Rozwiązania te wykazują znaczące zalety w porównaniu ze wspomnianymi powyżej implementacjami tablicowymi [4], należą do nich relatywnie niewielka zajętość zasobów oraz spora szybkość pracy przy właściwie dobranej funkcji aproksymującej. Rozważania te jednak dotyczyły obliczeń pojedynczej precyzji. Zastosowanie tych metod dla liczb o większej precyzji (większej szerokości argumentu realizowanej funkcji) narzuca jednak konieczność stosowania wielomianów wyższego stopnia, co z kolej związane jest z większą liczbą mnożeń. W tym miejscu należy podkreślić, że wraz z wzrastającą precyzją obliczeń rząd wielomianu gwałtownie wzrasta, wzrasta również szerokość układów mnożących, co pociąga za sobą gwałtowny wzrost zajmowanych zasobów.

Alternatywnym podejściem dla aproksymacji wielomianowej funkcji jest podzielenie obszaru aproksymowanego na mniejsze przedziały [5] i stosowanie w ich ramach wielomianów niższego stopnia (rys. 2).



Rys. 2. Różne sposoby realizacji aproksymacji wielomianowej
Fig. 2. Different approaches to polynomial approximation

Wzrastająca liczba przedziałów powoduje, że zmniejsza się rząd stosowanego wielomianu (zmniejsza się również ilość układów mnożących) kosztem wzrostu wielkości pojedynczej pamięci.

Zauważyć, więc można, że dla większych precyzji danych wejściowych (np. 64 bit) metoda aproksymacji wielomianowej traci swoje podstawowe zalety. Rozwiązanie tablicowe jest również obarczone istotnymi mankamentami w postaci bardzo dużej zajętości zasobów, co zostało uwypuklone na przykładzie pierwszej realizacji przedstawionej na rys. 2.

Wnioski płynące z powyższych obserwacji w naturalny sposób skłaniają do zastosowania metody alternatywnej, łączącej zalety dwóch wymienionych rozwiązań. Są to rozwiązania mieszane tablicowo-aproksymacyjne. W przeszłości zrealizowano wiele implementacji metod mieszanych obliczania funkcji elementarnych [6]. Dotyczyły one jednak w znakomitej większości operacji na danych zmiennoprzecinkowych, co najwyżej 32-bitowej precyzji [7]. Jako że metody mieszane obliczania funkcji elementarnych w standardzie liczb zmiennoprzecinkowych zostały najpierw zaimplementowane dla procesorów ogólnego przeznaczenia [8], pierwsze rozwiązania sprzętowe bardzo wyrazie czerpią z algorytmów softwarowych. Takie podejście nie wykorzystuje potencjału sprzętowego środowiska, tkwiącego w równoległości pracy i większych dostępnych zasobach logicznych. Dlatego obserwuje się obecnie prace mające na celu stworzenie szybkiej implementacji metody mieszanej obliczania funkcji exp() [9] oraz innych funkcji elementarnych. Jest to uzasadnione ogromnym potencjałem nowoczesnych układów programowalnych FPGA.

3. Proponowany moduł

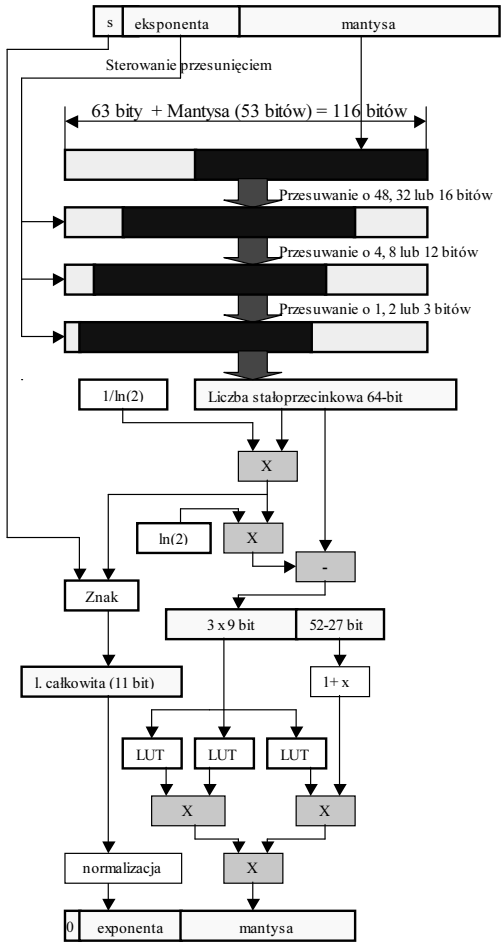
Analizując architekturę z rys. 1 można ją w prosty sposób ulepszyć poprzez zastosowanie większej pamięci LUT. Na przykład zamiast stosowania pamięci LUT o szerokości magistrali adresowej 1 można zastosować pamięć LUT o szerokości 4 (powszechnie stosowane w układach FPGA) dzięki temu liczba układów mnożących zmniejszy się z 63 do 15. Zwiększenie wielkości pamięci LUT nie powoduje praktycznie żadnych negatywnych konsekwencji. Oczywiste jest, że dalsze zwiększanie wielkości pamięci LUT powoduje zmniejszenie liczby wykonywanych operacji mnożenia kosztem jednak większej pamięci LUT, której zasoby zaczynają również gwałtownie wzrastać. Analiza zajmowanych zasobów przez pamięć LUT oraz układów mnożących doprowadziła do wniosku, że najlepszym rozwiązaniem będzie użycie dużych pamięci blokowych Block RAM (BRAM) znajdujących się w układach FPGA.

Na uwagę zasługuje podejście do liczb ujemnych argumentu wejściowego. W formacie znak-moduł (standard dla mantysy liczby zmiennoprzecinkowej), znak przenosi się na wszystkie pamięci LUT, co powoduje zwiększenie szerokości magistrali adresowej o jeden bit. Dlatego lepszym rozwiązaniem, zastosowanym w naszym rozwiązaniu jest konwersja do liczby w formacie uzupełnień do dwóch dzięki czemu, gdzie znak liczby ogranicza się tylko do najstarszego bitu [10]. Dzięki temu znak liczby jest uwzględniany tylko dla części całkowitej danej wejściowej.

Dodatkowe oszczędności można osiągnąć poprzez zastąpienie operacji mnożenia zmiennoprzecinkowego mnożeniem ze stałym przecinkiem. Jest to możliwe ze względu na fakt, że wydzielona część całkowita danej wejściowej jest rozpatrywana osobno. Ponadto dane wejściowe o wadze mniejszej niż około 2^{-60} mają znikomy wpływ na wynik końcowy i mogą być pominięte podczas obliczeń.

Proponowany moduł obliczeniowy składa się z następujących elementów (schemat blokowy przedstawiony został na rys. 4):

- logiki sprawdzania stanów wyjątkowych (inf, NaN) oraz konwersji danych wejściowych do wewnętrznego stałoprzecinkowego standardu zapisu liczb,
- logiki separującej część całkowitą od ułamkowej liczby, łącznie z migracją znaku do części całkowej
- tablic LUT przechowujących cząstkowe wartości exp(x),
- logiki normalizacji wyniku do standardu IEEE-754.



Rys. 3. Architektura proponowanego modułu obliczania funkcji exp()
Fig. 3. Architecture of the described module

4. Wyniki implementacji

Wyniki zamieszczone w tab. 1 nie uwzględniają mechanizmu potokowości, który będzie zaimplementowany w końcowej wersji modułu sprzętowego, gdy zakończony zostanie etap badań nad numerycznymi aspektami architektury modułu (oczekiwana dokładność obliczeń, wielkości tablic LUT, rząd wielomianu aproksymującego, wielkość zajmowanych zasobów).

Tab. 1. Wyniki implementacji modułu exp() bez mechanizmu potokowości oraz procentowa zajętość układu Xilinx Virtex-4 LX200
Tab.1. Implementation results of the exp() module on Xilinx Virtex-4 LX200, pipeline mechanism has not been implemented

Rodzaj	# 4-wej LUT	# przerzutników	# 18-Kb BRAM	# DSP48
Bez logiki DSP48	13375 (7.5%)	105 (0.06%)	6 (1.8%)	0
Z logiką DSP48	1293 (0.73%)	105 (0.06%)	6 (1.8%)	71 (74%)

W naszych badaniach korzystamy z komputera o dużej mocy obliczeniowej Altix 4700 firmy SGI. Przedstawiony projekt zostanie docelowo zaimplementowany na karcie SGI RASC RC100 Blade jako element większego systemu obliczeniowego. SGI RASC RC100 Blade został wyposażony w dwa układy typu Xilinx Virtex-4 LX200, dlatego zamieszono poniżej wyniki implementacji dotyczą właśnie tego układu FPGA.

Wykorzystanie bloków DSP48 wiąże się z użyciem wbudowanych mnożarek 18x18 będących ich integralna częścią. Takie podejście zmniejsza wyraźnie liczbę zaangażowanych pamięci LUT, co zostało przedstawione w powyżej tabeli. Jednakże liczba wbudowanych modułów mnożących jest stosunkowo niewielka i tylko jeden moduł exp() może być zaimplementowany z użyciem tych układów. Dlatego w przypadku użycia większej liczby modułów exp() konieczne jest użycie logiki ogólnego przeznaczenia (pamięci LUT) w ramach układu mnożącego.

5. Podsumowanie

W artykule przedstawiono architekturę sprzętowego modułu obliczania funkcji exp() o podwójnej precyzji. Należy podkreślić, że większość implementacji funkcji *exp* ograniczała się do pojedynczej precyzji. Zwiększenie precyzji obliczeń wymagało nie tylko zwiększenia dokładności poszczególnych operacji składowych, ale również rozważenia różnych algorytmów implementacji. Wybrano algorytm mieszany: 27-bitów najstarszych mantysy wejściowej jest obliczana za pomocą metody tablicowej. Natomiast mniej znaczące bity są obliczane za pomocą rozwinięcia w szereg Taylora. Ponieważ dana wejściowa jest bardzo mała (mniejsza niż 2^{-27}) szereg jest bardzo szybko zbieżny, wymaga tylko pierwszego rozwinięcia $e^x = 1 + x$.

Wyniki implementacji pokazują, że zasoby współczesnych układów FPGA są wystarczające do wykonywania operacji eksponenty o podwójnej precyzji obliczeń, co więcej możliwe jest umieszczenie w jednym układzie FPGA więcej niż 10 tego typu modułów obliczeniowych.

Praca finansowana ze środków budżetowych na naukę.

6. Literatura

[1] Jamro E. Parameterised automated generation of convolvers implemented in FPGAs, Ph.D. Thesis, University of Mining and Metallurgy (AGH), Kraków, Poland, June 2001.
[2] Stanek S., Benjegerdes T., Reconfigurable Computing for High Performance Technical Computing, Scalable Computing Lab, Ames Laboratory, Ames, IA 50010.
[3] Jeremie Detrey, Florent de Dinechin, Second Order Function Approximation Using a Single Multiplication on FPGAs, FPL 2004: Field-Programmable Logic and Applications Antwerp, 30 Aug. - 1 Sep. 2004, pp. 221-230.
[4] Detrey J., de Dinechin F, Table-based polynomials for fast hardware function evaluation, 16th IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'05), Samos, Greece, July 2005, pp. 328-333.
[5] Oskar Mencer, Wayne Luk, Dong-U Lee, Altaf Abdul Gaffar, Optimizing Hardware Function Evaluation, IEEE Transactions On Computers, Volume 54 , Issue 12 (December 2005) pp. 1520 – 1531.
[6] Doss C.C., Riley R.L., Jr., FPGA-Based Implementation of a Robust IEEE-754 Exponential Unit, 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04), pp. 229- 238.
[7] Bui H.T., Tahar S., Design and Synthesis of an IEEE-754 Exponential Function, IEEE Canadian Conference on Electrical and Computer Engineering Shaw Conference Center, Edmonton, Alberta, Canada May 9-12 1999, pp. 450-455 vol.1.
[8] Tang P., Table-Driven Implementation of the Exponential Function in IEEE Floating-Point Arithmetic, ACM Transactions on Mathematical Software (TOMS), Volume 15 , Issue 2 (June 1989), pp. 144 – 157.
[9] Detrey J., de Dinechin F., A parameterized foating-point exponential function for FPGAs, IEEE International Conference FPT'05, Singapore, December 2005, pp. 27-34.
[10] Wiatr K., Jamro E. Constant Coefficient Multiplication in FPGA Structures, Proc. of the IEEE Int. Conf. Euromicro, Maastricht, The Netherlands, Sep. 5-7, 2000, Vol. I, pp. 252-259.

Artykuł recenzowany