Dominik BUDYN [1], **Paweł RUSSEK** [1,2], Kazimierz WIATR [1,2]
[1] KATEDRA ELEKTRONIKI AGH, Al. Mickiewicz 30, 30-059 Kraków
[2] ACK CYFRONET AGH, Nawojka 11, 30-950, Kraków

Porównanie wydajności języków projektowania na przykładzie języka Mitrion-C oraz VHDL dla sprzętowego procesora CORDIC

Inż. Dominik BUDYN

W roku 2010 odbywał praktykę w ACK Cyfronet AGH, gdzie zajmował się tematyką akceleracji obliczeń naukowo-technicznych metodami sprzętowymi przy wykorzystaniu układów FPGA i języków wysokiego poziomu. Tytuł inżyniera uzyskał w 2011 roku w Katedrze Elektroniki AGH.. Aktualnie jest studentem studiów 2 stopnia na kierunku Elektronika i Telekomunikacja wydziału EAIiE na AGH.

e-mail: budyn.dominik@gmail.com

Dr inż. Paweł RUSSEK

Ukończył studia na wydziale Elektrotechniki, Automatyki i Elektroniki AGH Kraków (1994), dr nauk technicznych (2003). Jest adiunktem w Katedrze Elektroniki AGH i pracownikiem ACK Cyfronet. Prowadzone prace badawcze dotyczą sprzętowej akceleracji obliczeń przy pomocy architektur dedykowanych, zagadnień realizacji obliczeń przy użyciu rekonfigurowalnego sprzętu oraz wykorzystania układów reprogramowalnych w obliczeniach naukowych i technicznych wielkiej skali.

e-mail: russek@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

Narzędzia do projektowania bazujące na opisie HLL są już powszechnie dostępne dla projektantów struktur rekonfigurowalnych. Ciągle jednak, problemem jest wydajność osiągana przez dostępne rozwiązania. Aktualne i potrzebne jest więc porównywanie rozwiązań i poszukiwanie tych, które w określonych zastosowaniach sprawdzają się najlepiej. Artykuł porównuje dwie realizacje potokowego algorytmu CORDIC. Autorzy dzielą się swoimi wynikami oraz wnioskami i spostrzeżeniami, które powstały w toku realizacji obu implementacji.

Słowa kluczowe: języki opisu sprzętu, CORDIC, wydajność sprzętu, Mitrion-C, VHDL.

Performance comparison of hardware languages based on Mitrion-C and VHDL case study for CORDIC algorithm

Abstract

A design of hardware architectures using high level description languages becomes more and more popular in common engineering practice regarding science and technology. Design entry tools that accept a hardware description similar in syntax to ANSI C are commonly avaliable for designers of reconfigurable structures. Hovewer, despite maturity of those tools, performance is still a problem if compared to RTL descriptions which can be entered if languages such as Verilog and VHDL are used. Thus, comparing and evaluating the mentioned styles of hardware programming seems to be necessary and up-to-date. That can lead to a common knowledge what tools and languages are best for particular purposes. This paper presents a comparison of two implementaions of a CORDIC algorithm which were performed on the SGI RASC reconfigurable platform. The implementations were described both in VHDL and a high level style hardware language: Mitrion-C. The authors present the results, remarks and conclusions which arose during the process of creation of both implementations.

Keywords: hardware description languages, CORDIC, hardware performance, Mitrion-C, VHDL.

1. Wprowadzenie

Projektowanie układów cyfrowych z wykorzystaniem języków programowania wysokiego poziomu HLL (ang. High Level Languages) zyskuje na znaczeniu. Narzędzia te pozwalają zredukować problem niewystarczającej wydajności zespołów projektowych, który stał się problemem wobec ciągłego wzrostu pojemności dostępnych cyfrowych układów rekonfigurowalnych.

Typowo projekty wykonane przy pomocy HLL cechuje mniejsza wydajność niż tych wykonanych tradycyjnie przy zastosowaniu języków HDL. Jednak w zamian znacznie skracają one czas realizacji projektu.

Język Mitrion-C [1] stworzony przez firmę Mitrionics, jest jednym z oferowanych obecnie komercyjnie języków wysokiego poziomu. Pomyślany on został do programowania platform rekonfigurowalnych, służących do realizacji obliczeń wielkiej skali na potrzeby nauki i techniki. Są to tak zwane platformy HPRC (ang. High Performance Reconfigurable Computing).

Procesor bazujący na FPGA działa przy znacznie niższej częstotliwości zegara niż procesor ogólnego zastosowania, jednak dzięki zrównolegleniu operacji efektywny czas wykonywania algorytmu może ulec skróceniu. W związku z tym, akceleracja obliczeń dostępna jest wyłącznie dla algorytmów, w których możliwe jest zrównoleglenie obliczeń. Jednym z takich algorytmów jest algorytm CORDIC, którego implementację w języku Mitrion-C przedstawiono w niniejszej pracy. Głównym celem pracy było oszacowanie wydajności języka, HLL w stosunku do tradycyjnej metody projektowania. W tym celu stworzono przy pomocy języka Mitrion-C moduł CORDIC, a następnie porównano jego parametry do implementacji tego samego algorytmu wykonanego w języku VHDL.

2. Algorytm CORDIC

Algorytm CORDIC jest szeroko opisany w literaturze, np. w [2]. Również sposób jego realizacji w FPGA był szeroko opisywany [3, 4]. Wyróżniane są dwie wersje algorytmu CORDIC: rotacyjna i wektorowa. W niniejszej pracy zastosowano wersję rotacyjną. Tryb ten wykorzystywany jest między innymi do obliczania wartości funkcji sinus i cosinus. Iteracyjny algorytm CORDIC można opisać następującymi równaniami:

$$\begin{aligned} X_{i+1} &= X_i - d_i * Y_i * 2^{-i} \\ Y_{i+1} &= Y_i + d_i * X_i * 2^{-i} \\ \alpha_{i+1} &= \alpha_i - d_i * \tan^{-1}\left(2^{-i}\right) \\ d_{i+1} &= sign(\alpha_i) \end{aligned} \tag{1}$$

gdzie X_i, Y_i, α_i, d_i wartości wyliczane w i-tym kroku algorytmu.

Dodatkowo zakładamy wartości początkowe: X_0=1 Y_0=0 i α_0=α (wartość kąta, dla którego, wyliczany jest sinus i cosinus). Po n krokach pracy algorytmu cos(α)=X_n i sin(α)=Y_n.

W opisywanej implementacji, dla opisywanego iteracyjnego algorytmu CORDIC, zastosowano równoległość na poziomie realizowanych instrukcji (ILP) oraz przetwarzanie potokowe dla serii niezależnych od siebie danych wejściowych.

3. Język Mitrion-C

Filozofia, jaka przyświecała twórcom języka Mitrion-C polegała na dążeniu do stworzeniu narzędzia, które udostępni technikę akceleracji obliczeń szerokiemu gronu programistów softwaru. Zasada jest konsekwentnie realizowana w podręcznikach Mitrion-C, gdzie niezbyt dociekliwy użytkownik, ucząc się narzędzia, może nie być świadomy, jaka jest docelowa platforma sprzętowa, na której, będzie uruchamiany jego projekt. Nie przeszkadza to jednak w osiąganiu przyśpieszeń, a to dzięki wpojeniu zasad stosowania kilku prostych reguł m.in.: wykorzystywanie równoległości i potokowość, operowanie na typie danych użytkownika i stosowanie zasady unikania wzajemnej zależności danych.

Język Mitrion-C nie ma nic wspólnego z językiem ANSI C, choć jest językiem opisu wysokiego poziomu. Cechą charakterystyczna języka Mitrion-C jest to, że wszystkie realizowane w programie instrukcje są wykonywane równocześnie lub kolejno, jeżeli istnieje zależność pomiędzy poszczególnymi instrukcjami. Każda instrukcja jest realizowana tak szybko jak tylko są dostępne jej argumenty (ASAP). Zastosowano tu filozofię kolejkowania bez ograniczeń sprzętowych. Jeżeli opisana struktura nie mieści się w FPGA, programista musi zmienić sposób opisu. Ciekawą cechą jest to, że każda instrukcja, w tym pętle i instrukcje warunkowe zwracają wartość. Efektem działania narzędzi Mitrion jest stworzenie wirtualnego procesora MPV (Mitrion Virtual Processor).

Innym wyznacznikiem Mitrion-C spośród znanych języków programowania sprzętu wysokiego poziomu jest zasada, że programista sam ma wiedzieć jak dany algorytm ma być zrealizowany i ma mieć możliwość łatwego, szybkiego i przejrzystego opisu swojego pomysłu w odpowiednim języku. Programista sam decyduje o stopniu równoległości i liczbie elementów funkcjonalnych. Optymalizacja jest raczej bardzo niskopoziomowa. Narzędzie samo z siebie nie wprowadzi potoku czy równoległej realizacji danych. Dużą pomocą dla programisty ze strony narzędzia jest automatyczne dobieranie typu wyniku na podstawie typu danych wejściowych i rodzaju operacji.

Procesor MPV zawsze pracuje z częstotliwością 100MHz, nawet, jeżeli częstotliwość graniczna otrzymana dla projektu jest większa. Częstotliwość ta została przez twórców języka ustalona na sztywno, co wynika z pewnego rodzaju asekuracji firmy Mitrionics, która gwarantuje, że wykonane przy pomocy narzędzia Mitrion projekty będą zawsze działały zgodnie z symulacją. Jeżeli projekt nie działa, firma na własny koszt zajmuje się jego uruchomieniem! Opis w języku Mitrion-C jest automatycznie zamieniany przez narzędzie Mitrion na opis w języku VHDL, który następnie musi zostać skompilowany i zaimplementowany w układzie FPGA przez narzędzia dostarczane przez firmę Xilinx: ISE Design Suite.

4. Implementacja

Implementację algorytmu przeprowadzono na platformie RASC [5], która jest ogólnie dostępna w ACK Cyfronet dla szerokiego grona użytkowników. Platforma RASC jest wyposażona w układ FPGA Virtex4LX200 firmy Xilinx. Kompilator Mitrion-C jest również dostępny w ACK.

Pierwszą decyzją podjętą podczas implementowania algorytmu był wybór sposobu wymiany danych pomiędzy procesorem wirtualnym i hostem. Platforma RASC umożliwia komunikacją za pośrednictwem bloków pamięci RAM znajdujących się w układzie FPGA. Słowo w pamięci jest 128 bitowe. Pamięć przedstawiona jest jako typ tablicowy składający się z 0x100000 takich słów. Algorytm operuje na słowie 32-bitowym i w wyniku jego działania otrzymujemy wynik składający się z dwóch słów 32-bitowych, więc podczas działania algorytmu wykorzystywana jest ¼ pamięci wejściowej i ½ pamięci wyjściowej. Do reprezentacji liczb wybrano format stałoprzecinkowy 32 bitowy.

Do syntezy wykorzystano dodatkowo narzędzie ISE w wersji 9.2. Kod programu dostępny jest na stronach internetowych [6].

5. Wyniki

Wstępna analiza wykorzystywanych przez projekt zasobów dokonywana jest już podczas kompilacji procesora MPV przy pomocy narzędzia Mitrion do kodu VHDL. Dokładniejsze wyniki dotyczące wykorzystania zasobów i wydajności można uzyskać po dokonaniu syntezy i implementacji przy pomocy narzędzi firmy Xilinx. Tak jak już wspominano w pracy dokonano również syntezy algorytmu opisanego w języku VHDL, aby móc ocenić wydajność języka Mitrion-C.

Raport szczegółowy dla syntezy opisu wykonanego w języku Mitrion-C przedstawiony został w tabeli 1. Jest to raport z ostatecznych wyników, uzyskany po dokonaniu przez narzędzia ISE syntezy i implementacji.

Tab. 1. Wykorzystanie zasobów przez MPV dla algorytmu CORDIC
Tab. 1. MPV resources utilization for CORDIC algorithm

Lp.	Typ zasobu	Wykorzystanie
1	flip-flops	30 737 (17%)
2	slices	23 861 (26%)
3	BRAMs	12 (3%)

Analogiczny raport został również wygenerowany dla algorytmu opisanego w języku VHDL. Tabela 2 zawiera porównanie między wynikami syntezy algorytmu opisanego w obu rozważanych językach.

Tab. 2. Porównanie wykorzystania zasobów dla algorytmu CORDIC przy zastosowaniu VHDL i Mitrion-C
Tab. 2. Comparison of resources utilization for CORDIC algorithm when VHDL and Mitrion-C are used

Lp.	Typ zasobu	Wykorzystanie	
		Mitrion-C	VHDL
1	add/sub (32 bit)	202	90
2	flip-flops	23 861	2 977

Różnice pomiędzy wykorzystaniem układu FPGA przez algorytm opisany w językach Mitrion-C i VHDL jest bardzo duża. Ma to miejsce szczególnie w przypadku liczby przerzutników (7-krotna). Różnica w wykorzystaniu układów arytmetycznych nie jest już tak dramatyczna i jest trochę ponad 2 krotna. Ta dwukrotna różnica jest faktycznym wskaźnikiem różnicy wydajności VHDL i Mitrion-C. Poniżej uzasadniamy dlaczego.

Ogromna różnica liczby przerzutników wynika głównie z cechy kompilatora Mitrion. Polega ona na bardzo agresywnym stosowaniu potokowości. Oznacza to, że stosuje się rejestry potokowe, wychodząc z założenia, że nie wpływają one na wydajność układu przy potokowej realizacji operacji na dużym zbiorze danych wejściowych. Rejestry potokowe wydłużają co prawda opóźnienie potokowe, ale znacznie ułatwiają pracę narzędziom *place&route*. Trzeba dodać, że w HPRC potokowe przetwarzanie zbioru niezależnych danych wejściowych to sytuacja bardzo typowa.

Liczba przerzutników jest znacznie większa dla algorytmu opisanego w języku Mitrion-C, ponieważ – mimo niższej częstotliwości zegara niż dla opisu w języku VHDL – zastosowana została większa liczba poziomów potoku. Każdy poziom potoku potrzebuje pełnego zestawu rejestrów, aby przechować wszystkie zmienne. W przypadku kodu VHDL potok podzielony jest na 32 etapy, zaś w języku Mitrion-C na 95 etapów. Jak wiadomo w układach FPGA, zwiększanie poziomów potokowości jest osiągane „za darmo”, ponieważ rejestry skojarzone z logiką kom-

binacyjną (LUT) są i tak dostępne, a ich niewykorzystanie nie zmniejsza wykorzystania zasobów *slice*.

Dodatkowo trzeba przyznać, że wynik pracy kompilatora Mitrion nie jest optymalny. Przykładowo, do porównania aktualnej zawartości akumulatora kąta z zerem wykorzystywany jest komparator 32 bitowy, podczas gdy wystarczy sprawdzić znak akumulatora kąta. Nie jest również konieczne wykorzystanie rejestrów przesuwnych. Wartości są w danej iteracji przesuwane zawsze o stałą liczbę bitów, więc rejestr przesuwny może zostać zastąpiony przez odpowiednie połączenia pomiędzy rejestrami, tak jak jest to robione w przypadku opisu algorytmu w języku VHDL.

Większa liczba układów dodających i odejmujących również wynika z zastosowania w kodzie obliczeń warunkowych. Układ opisany w języku VHDL potrzebuje w każdej iteracji jednego układu add/sub, sterowanego bitem znaku akumulatora kąta dla każdej zmiennej. Układ opisany w języku Mitrion-C potrzebuje po jednym układzie dodającym i jednym układzie odejmującym dla każdej zmiennej w każdej iteracji. Szerokość tych układów (33 bity) wynika z interferencji typów. Dodatkowo, w związku z obliczeniami warunkowymi konieczne było zastosowanie dwukrotnie większej liczby rejestrów. Pozostałe rejestry wykorzystywane są jako bufory przy odczycie, zapisie i przesyłaniu danych.

Zmienne przechowujące wynik operacji dodawania i odejmowania nie mają w programie zdefiniowanej przez programistę precyzji. W związku z czym, kompilator dobrał zakres tak, żeby nie został utracony żaden bit wyniku. Optymalizacja wykorzystania tych zasobów z poziomu języka Mitrion-C, poza jawnym zadeklarowaniem precyzji zmiennych, nie jest możliwa.

6. Wydajność

W celu określenia wydajności oceniano przepustowość procesorów sprzętowych MPV i VHDL. Ocena nie uwzględniania czasu transferu danych pomiędzy platformą RASC i komputerem *host*. Rozważono więc teoretyczną przepustowość procesora sprzętowego.

Przepustowość algorytmu ocenić można na podstawie symulacji programu napisanego w języku Mitrion-C. Symulacja pozwala uzyskać informację o liczbie cykli zegara potrzebnych do wykonania obliczenia na pojedynczym zestawie danych wejściowych. Wynik dla MPV to 1 cykl zegara na jeden zestaw danych wejściowych przy częstotliwości taktowania 100MHz.

W podobny sposób można wyznaczyć czas wykonywania algorytmu opisanego w języku VHDL. Maksymalna częstotliwość pracy zegara wynosi w tym przypadku 192 MHz. W tym przypadku również efektywnie przetwarzana jest jedna dana wejściowa na jeden cykl zegara.

7. Czas projektowania

Porównanie czasu potrzebnego na wykonanie projektu jest problematyczne, ponieważ opisywana implementacja algorytmu CORDIC była pierwszym projektem wykonywanym w tym języku. W związku z tym znaczna część czasu spędzonego na pracy nad projektem wykorzystana była na naukę języka oraz związanego z nim zestawu narzędzi do programowania i symulacji. Język Mitrion-C wywodzi się z języka C, lecz różnice między nimi są na tyle duże, że konieczne jest spędzenie czasu na zapoznanie się z cechami charakterystycznymi składni. Mimo to, wykonanie opisu algorytmu w języku Mitrion-C było łatwiejsze oraz mniej czasochłonne, niż opis tego samego algorytmu w języku VHDL. W przypadku kolejnych algorytmów różnica byłaby jeszcze bardziej widoczna. Ze względu na możliwość zastosowania metod wyższego poziomu kod utworzony w języku Mitrion-C jest bardziej zwięzły, co bezpośrednio przekłada się na łatwość jego utrzymywania.

Całość kodu opisującego algorytm CORDIC w języku Mitrion-C, po usunięciu komentarzy, zawiera się w niespełna 50 liniach kodu. Zwięzłość kodu oraz sprawne narzędzia symulacyjne pozwalają skrócić czas od rozpoczęcia projektu do uruchomienia go na platformie docelowej. Kod pisany w języku Mitrion-C charakteryzuje się również większą elastycznością i większą łatwością modyfikacji niż kod napisany w języku VHDL. Uruchomienie zmodyfikowanego kodu wiąże się z koniecznością wykonania ponownie czasochłonnego procesu syntezy układu, jednak samo wprowadzenie zmian jest znacznie mniej pracochłonne niż w języku VHDL.

8. Wnioski

Porównania wydajności oraz zasobów sprzętowych wykorzystywanych przez algorytmy opisane z wykorzystaniem języków Mitrion-C i VHDL świadczy jednoznacznie na korzyść języka VHDL. Jego wydajność jest prawie 2 razy większa, a zajmowane zasoby kilkukrotnie mniejsze. Oznacza to, że język Mitrion-C ani inne języki programowania wysokiego poziomu na razie nie wyprą tradycyjnych języków opisu sprzętu, takich jak VHDL, z zastosowań wymagających wysokiej wydajności lub uruchamianych na platformie o ograniczonych zasobach. Jak jednak wspominano, gdyby nie ustalona na stałe częstotliwość MPV różnica nie byłaby tak duża.

Języki programowania wysokiego poziomu posiadają jednak liczne zalety, które pozwalają na wykorzystanie ich w procesie tworzenie dedykowanych architektur sprzętowych służących do akceleracji obliczeń. Przede wszystkim czas projektowania algorytmu jest znacznie krótszy niż przy użyciu języków niskiego poziomu. Łatwiejsze jest również wprowadzanie zmian, w przypadku, gdy zaprojektowana architektura nie spełnia wymagań lub zajmuje zbyt dużą część zasobów układu. Języki wysokiego poziomu umożliwiają również przenoszenie sprzętu pomiędzy różnymi platformami rekonfigurowalnymi lub ewentualnymi nowymi wersjami platform, które nie będą kompatybilne wstecz. Do uruchomienia algorytmu opisanego w języku wysokiego poziomu konieczne jest ponowne wykonanie procesu syntezy oraz wprowadzenie zmian w kodzie programu, związanych z cechami charakterystycznymi dla danej platformy, takimi jak organizacja pamięci czy sposób przekazywania danych wejściowych.

Ogólnie można podsumować, że Mitrion-C to dobre narzędzie. W szczególności przydatne do szybkiej weryfikacji w FPGA nowych pomysłów, które w przypadku sukcesu warto przepisać ręcznie w HDL, dzięki czemu można liczyć w pierwszej kolejności na zyski w liczbie zużytych zasobów.

Praca powstała ze środków na naukę Narodowego Centrum Badań i Rozwoju w 2011r, w ramach projektu SYNAT.

9. Literatura

[1] Mitrionics Inc. Mitrion User's Guide [Online]. Available: www.mitrionics.com

[2] Jack E. Volder: The CORDIC Trigonometric Computing Technique, IRE Transactions on Electronic Computers, pp330-334, September 1959.

[3] Duprat J., Muller J.M.: The CORDIC Algorithm: New Results for Fast VLSI Implementation, IEEE Transactions on Computers, pp. 168-178, February, 1993.

[4] Ray Andraka: A survey of CORDIC algorithms for FPGAs, Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Feb. 22-24, 1998, Monterey, CA. pp. 191-200.

[5] Silicon Graphics Inc., Mountain View, CA, SGI RASC Technology Datasheet, 2005.

[6] http://www.cyfronet.pl/uslugi_obliczeniowe/?a=zao/PlatformaRekonfigurowalna/

otrzymano / received: 13.05.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*