Paweł POCZEKAJŁO

Politechnika Koszalińska, Wydział Elektroniki i Informatyki, Katedra Systemów CPS

doi:10.15199/48.2018.09.06

Analiza wybranych metod realizacji sprzętowej rotatorów Givensa w układzie FPGA

Streszczenie. *W artykule przedstawiono analizę wybranych metod sprzętowej realizacji rotatorów Givensa. Jako bazę sprzętową wybrano układ FPGA Altera Cyclone 5 SX SoC. Dokonano również pomiaru szeregu parametrów sprzętowych oraz sprawdzono dokładność działania, następnie porównano wyniki w celu wybrania najlepszej realizacji. Badania zostały zrealizowane w ramach prac nad ortogonalnymi filtrami potokowymi realizowanymi za pomocą rotatorów Givensa.*

Abstract. *In the paper, analysis of selected methods of hardware implementation of Givens rotators is presented . The author use the FPGA chip Altera Cyclone 5 SX SoC. Hardware parameters of structures are measure. Also accuracy of operations are analysed. Results are compared and selected the best implementation. The research are realisation as part of the work on orthogonal filters implemented with Givens rotators. (**Analysis of methods to hardware realisation of Givens rotation in FPGA chip**).*

Słowa kluczowe: rotator Givensa, realizacja, implementacja, FPGA, algorytm CORDIC, bloki DSP, filtry rotatorowe, analiza.
Keywords: Givens rotation, realisation, implementation, FPGA, CORDIC algorithm, DSP block, rotation filter, analysis.

Wprowadzenie

W ostatnich latach znacząco wzrosła popularność wszelkiego rodzaju akceleracji sprzętowej stosowanej w programach i środowiskach realizujących duże ilości obliczeń [1, 2, 3]. Najpopularniejsze architektury tego rodzaju to systemy CUDA (NVidia) oraz OpenCL. Na rynku dostępne są dedykowane rozwiązania sprzętowe np. NVIDIA Quadro, czy NVIDIA Tesla. Coraz większą popularność zdobywają również rozwiązania oparte na układach FPGA, np. Terasic DE5-Net i Terasic DE10-Standard. Środowiska obliczeniowe jak Scilab i Matlab, dzięki dedykowanym bibliotekom, również mają możliwość wspierania akceleracji sprzętowej przy realizacji różnych obliczeń.

Układy FPGA, ze względu na możliwości, często wykorzystywane są w systemach cyfrowego przetwarzania sygnałów (CPS). Bezpośrednia realizacja filtrów o skończonej i nieskończonej odpowiedzi impulsowej (FIR i IIR) nie stanowi żadnego problemu [4]. Dodatkowo producenci poszczególnych układów dostarczają gotowe rozwiązania programowe znacznie ułatwiające implementację takich systemów [5]. Przedstawiono również koncepcję realizacji ortogonalnych filtrów cyfrowych za pomocą rotatorów Givensa [6, 7]. Taka realizacja sprzętowa wybranych filtrów FIR/IIR pozwala na otrzymanie systemów potokowych, dodatkowo wskazuje się na poprawę wybranych parametrów [8, 9]. Jednymi z istotniejszych są te związane bezpośrednio z implementacją w układzie FPGA (np. zajętość w docelowym układzie reprogramowalnym, maksymalne możliwe taktowanie). Niniejszy artykuł skupia się na rotatorze Givensa, który jest podstawowym elementem składowym takiej struktury. Omówiono wybrane metody realizacji sprzętowej, przedstawiając ich wady i zalety oraz prezentując podstawowe parametry, które zostały dobrane pod kątem realizacji filtrów rotatorowych.

Rotator Givensa

Rotator Givensa [10], jest to macierz którą w sposób ogólny można przedstawić następująco:

(1)
$$R_G = \begin{bmatrix} 1 & & & & & 0 \\ & \ddots & & & & \\ & & \cos(\alpha) & \cdots & -\sin(\alpha) & \\ & & \vdots & \ddots & \vdots & \\ & & \sin(\alpha) & \cdots & \cos(\alpha) & \\ & & & & & \ddots \\ 0 & & & & & 1 \end{bmatrix}$$

W poniższej pracy, zgodnie z zastosowaniami rotatorów Givensa w ortogonalnych filtrach potokowych [8, 9], jako rotator przyjęto następującą realizację:

(2)
$$\begin{bmatrix} v \\ t \end{bmatrix} = \boldsymbol{R}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos(\alpha) - y\sin(\alpha) \\ x\sin(\alpha) + y\cos(\alpha) \end{bmatrix}$$

gdzie: x, y – wartości wejściowe rotatora, v, t – wartości wyjściowe rotatora, α – kąt obroty rotatora.

Rotator (2) wykonuje obrót punktu o współrzędnych (x, y) wokół środka układu współrzędnych, gdzie (v, t) to współrzędne punktu po obrocie o kąt α. Zgodnie z publikacjami [6, 7], szeregowo połączone rotatory i bloki opóźniające mogą realizować niemal dowolny filtr FIR lub IIR. Sprzętowa realizacja takiego systemu jest niezwykle istotna, gdyż ma ona bezpośredni wpływ na parametry danego filtru. W kolejnym punkcie zaproponowano wybrane metody implementacji działania (2) z uwzględnieniem realizacji w układzie FPGA.

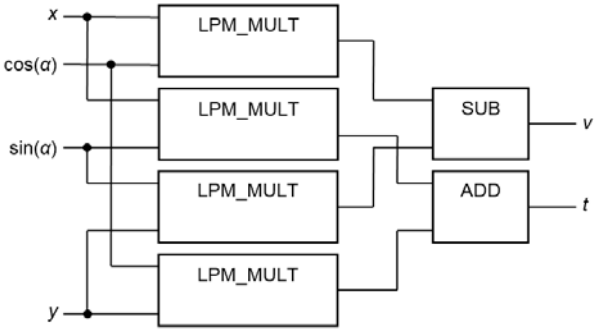
Sprzętowa realizacja rotatora

Jako układ sprzętowy zastosowany do realizacji rotatora wykorzystano zestaw uruchomieniowy Terasic SoCKit [11] z procesorem FPGA Altera Cyclone IV SX SoC (5CSXFC6D6F31C6N). Środowisko wykorzystane przy implementacji to Altera Quartus II 14.0 Web Edition. Dla (2) przyjęto, że parametry wejściowe każdej realizacji rotatora to x, y, cos(α) i sin(α), natomiast sygnały wyjściowe to v i t. Wszystkie współrzędne (x, y, v, t) są kodowane jako zmienne całkowite 8-bitowe ze znakiem zapisane w kodzie U2. Natomiast wartości funkcji trygonometrycznych (cos(α) i sin(α)) kodowane są jako zmienne stałoprzecinkowe 12-bitowe Q2.10 (2 bity całkowite i 10 bitów ułamkowych), ze znakiem w kodzie U2. Przedstawiona konfiguracja jest zgodna z [8, 9]. Poniżej zaprezentowano wybrane realizacje rotatorów Givensa.

Bezpośrednia realizacja na elementach logicznych

Bezpośrednia realizacja rotatora (2) opiera się na wykorzystaniu czterech bloków mnożących i dwóch sumatorów tak jak zaprezentowano to na rysunku 1. Kluczowymi działaniami są tu operacje mnożenia. Pierwsza proponowana metoda implementacji opiera się na

wykorzystaniu podstawowych elementów logicznych (bramki, przerzutniki). W praktyce struktura taka robi się dość skomplikowana i duża. W niniejszym opracowaniu oparto się o udostępnione przez firmę Altera biblioteki (IP Cores). Dostępny jest m.in. układ mnożący LPM_MULT, który można dowolnie skonfigurować (wielkość sygnałów wejściowych i wyjściowych, elementy użyte do implementacji, potokowość, itp.). Jak już wcześniej wspomniano, w tej realizacji wszystkie działania wykonywane są przy wykorzystaniu podstawowych elementów logicznych.



Rys.1. Schemat bezpośredniej struktury realizującej rotator Givensa

Bezpośrednia realizacja na blokach dsp

Bloki DSP dostępne w układzie FPGA, to wyspecjalizowane struktury sprzętowe pozwalające na bardzo szybkie realizowanie skomplikowanych działań i obliczeń, jak np. mnożenie, pierwiastkowanie, potęgowanie, wielowejściowe sumatory, dekodery/enkodery. Bloki te są w pełni programowalne i konfigurowalne, co zapewnia różne możliwość ich wykorzystania. Wykonanie działań (2) przy zastosowaniu dedykowanych bloków DSP, sprowadza się w praktyce do realizacja jak w punkcie 3.1. Jedyna różnica jest w konfiguracji elementu LPM_MULT, gdzie należy zaznaczyć, że przy kompilacji ma być realizowany właśnie na blokach DSP. Dodatkową zaletą w przypadku wykorzystanego procesora, jest możliwość realizacji dwóch niezależnych operacji mnożenia na jednym bloku DSP.

Algorytm cordic

W pierwotnej wersji algorytm iteracyjny CORDIC został zaproponowany w [12] jako łatwy do implementacji sprzętowej system pozwalający na iteracyjne wyznaczenie wartości funkcji trygonometrycznych. Zmodyfikowane wersje tego algorytmu są powszechnie wykorzystywane m.in. właśnie do realizacji obrotu punktu w układzie współrzędnych. Pojedyncza iteracja takiego algorytmu wygląda następująco:

$$a_i = \frac{a_{i-1}}{2}$$

$$s_i = \begin{cases} s_{i-1} + a_{i-1} \; dla \; s_{i-1} < 0 \\ s_{i-1} - a_{i-1} \; dla \; s_{i-1} \geq 0 \end{cases}$$

$$c_i = \begin{cases} c_{i-1} + a_{i-1} \; dla \; c_{i-1} < 0 \\ c_{i-1} - a_{i-1} \; dla \; c_{i-1} \geq 0 \end{cases}$$

$$v_i = \begin{cases} v_{i-1} + r_{i-1} \; dla \; c_{i-1} < 0 \; i \; s_{i-1} \geq 0 \\ v_{i-1} - r_{i-1} \; dla \; c_{i-1} \geq 0 \; i \; s_{i-1} < 0 \\ v_{i-1} - p_{i-1} \; dla \; c_{i-1} < 0 \; i \; s_{i-1} < 0 \\ v_{i-1} + p_{i-1} \; dla \; c_{i-1} \geq 0 \; i \; s_{i-1} \geq 0 \end{cases}$$

(3)

$$t_i = \begin{cases} t_{i-1} + p_{i-1} \; dla \; c_{i-1} \geq 0 \; i \; s_{i-1} < 0 \\ t_{i-1} - p_{i-1} \; dla \; c_{i-1} < 0 \; i \; s_{i-1} \geq 0 \\ t_{i-1} - r_{i-1} \; dla \; c_{i-1} < 0 \; i \; s_{i-1} < 0 \\ t_{i-1} + r_{i-1} \; dla \; c_{i-1} \geq 0 \; i \; s_{i-1} \geq 0 \end{cases}$$

gdzie: i = 1, 2, 3, … n, n – ilość iteracji algorytmu CORDIC, a_0 = 1, s_0 = sin(α), c_0 = cos(α), $r_0 = x+y$, $p_0 = x-y$, $v = v_n$, $t = t_n$.

Algorytm (3) jest iteracyjnie powtarzany przez n-cykli. W praktyce liczbę kolejnych iteracji warto ograniczyć do długości bitowej poszczególnych parametrów, tzn. jeśli argument jest n-bitowy, nie ma sensu wykonywanie większej liczby iteracji, gdyż w każdej kolejnej odpowiednie rejestry będą już wyzerowane. Dla opisywanego przypadku ograniczono się do 11 iteracji (tj. o jeden mniej niż reprezentacja bitowa argumentów sin(α) i cos(α)). Liczba iteracji w tym wypadku wynika też z opracowań [8] i [9]. Bloki realizujące kolejne iteracje zostały połączone szeregowo. Dzięki temu, w momencie gdy i-ty blok przelicza daną próbkę, poprzedni blok (i-1) przelicza już próbkę kolejną. Największym problemem takiego rozwiązania jest jego złożoność oraz zawiła analiza teoretyczna błędów kwantowania.

Badanie parametrów rotatorów

Ponieważ, niniejsze opracowanie skupia się na implementacji i badaniu samych rotatorów, mierzone będą jedynie wybrane parametry (niezwiązane bezpośrednio z realizacją całych filtrów). Sprawdzono m.in. dokładność działania rotatorów. Każda z realizacji przeliczyła identyczny zestaw losowych wektorów wejściowych (x, y). Natomiast wszystkie realizacje rotatora mają zadane identyczne wartości funkcji trygonometrycznych cos(α) i sin(α) dla losowo dobranego kąta obrotu. Wyniki pomiarów były rejestrowane w pamięci SDRAM oraz następnie odczytane do komputera i przeanalizowane w środowisku Scilab (porównano je do wyników wyznaczonych przy pełnej dostępnej precyzji). Wyznaczono również procentowo, jak duża ilość próbek wyjściowych różni się od wartości przewidywanej w reprezentacji całkowitej (zaokrąglone). Dodatkowo zmierzono parametry związane z implementacją sprzętową (np. zajętość w układzie FPGA, zastosowanie bloków DSP, maksymalna częstotliwość taktowania). Jest to istotne ze względu na to, że pojedynczy filtr potrafi składać się nawet z kilkunastu rotatorów, a liczba dostępnych elementów może być znacznie ograniczona (np. bloki DSP, których w wybranym układzie FPGA jest 112). Maksymalna częstotliwość taktowania, została zmierzona jako maksymalna częstotliwość, przy której dana realizacja zwraca wynik zgodny z oczekiwanym (nie uzyskujemy dodatkowych błędów). Do generowania przebiegu taktującego o częstotliwości większej niż 50MHz zastosowano pętle PLL. Parametry związane z realizacją w układzie FPGA zostały odczytane z raportu po kompilacji danego projektu w środowisku Altera Quartus. Wyniki opisanych pomiarów zostały zebrane w tabeli 1.

Podsumowanie

W przedstawionym artykule zaprezentowano analizę wybranych realizacji sprzętowych rotatorów Givensa. Przeprowadzone badania, zostały wykonane w kontekście wykorzystania rotatorów przy implementacji potokowych filtrów ortogonalnych. Zebrane wyniki przedstawiono w tabeli 1, gdzie zaznaczono najkorzystniejsze wartości dla każdego pomiaru. Realizacje bezpośrednie (na elementach logicznych i blokach DSP) są tożsame jeśli chodzi o generowane błędy.

Tabela 1. Wybrane parametry dla różnych realizacji rotatora Givensa

	METODA REALIZACJI ROTATORA		
	Bezpośrednia realizacja na elementach logicznych	Bezpośrednia realizacja na blokach DSP	Algorytm iteracyjny CORDIC
Liczba rejestrów	**189**	0	779
Liczba elementów logicznych (ALM's)	**205**	9	867
Liczba bloków DSP	0	2 (jeden blok wykonuje dwie operacje mnożenia)	0
Maks. częstotliwość taktowania	125MHz	**150MHz**	**150MHz**
Średnia błędu	0.0085343	0.0085343	**0.0016984**
Wariancja błędu	**0.0838450**	**0.0838450**	0.0919644
Odsetek błędów (pomiędzy wynikami, a oczekiwanymi wartościami całkowitymi)	**2,1%**	**2,1%**	8,1%

Warto tu zaznaczyć, że wszystkie występujące błędy, dla każdej realizacja mieściły się w zakresie od -1 do 1, więc można założyć, że wynikają one jedynie z kwantowania współczynników i skończonej precyzji obliczeń. Najistotniejszymi parametrami są maksymalna częstotliwość taktowania i zajętość struktury FPGA. Wadą algorytmu CORDIC jest bardzo duże zużycie zasobów procesora. Częściowo rekompensowane jest to osiągnięciem dość wysokiej częstotliwości taktowania. Identyczne taktowanie można osiągnąć dla rotatora opartego na blokach DSP, jednak ich dostępność w wybranych procesorach jest istotnym ograniczeniem. Teoretycznie najbardziej wygodne rozwiązanie to bezpośrednia realizacja rotatora na elementach logicznych, gdzie zużywa się stosunkowo niewielkie zasoby procesora, a jednocześnie maksymalna częstotliwość taktowania jest niewiele mniejsza. Dalsze etapy badań będą skupiały się nad wykorzystaniem bezpośrednich metod realizacji rotatora w implementacji filtrów ortogonalnych oraz porównaniu parametrów tak otrzymanych systemów CPS.

***Autorzy**: mgr inż. Paweł Poczekajło, Politechnika Koszalińska, Wydział Elektroniki i Informatyki, Katedra Systemów CPS, ul. Śniadeckich 2, 75-453 Koszalin.*

LITERATURA

[1] Iakymchuk T., Rosado-Muñoz A., Mompéan M.B., Víllora J.V.F., Osimiry E.O., Versatile Direct and Transpose Matrix Multiplication with Chained Operations: An Optimized Architecture Using Circulant Matrices, *IEEE Transactions on Computers*, 65 (2016), no. 11, 3470-3479

[2] Yang H., Ziavras S.G., Hu J., FPGA-based Vector Processing for Matrix Operations, *Information Technology, 2007. ITNG '07. Fourth International Conference on*, Las Vegas, (2007), 989-994

[3] Zhang Y., Shalabi Y.H., Jain R., Nagar K.K., Bakos J.D., FPGA vs. GPU for sparse matrix vector multiply, *2009 International Conference on Field-Programmable Technology*, Sydney, (2009), 255-262

[4] Paul A., Khan T.Z., Podder P., Hasan M.M., Ahmed T., Reconfigurable architecture design of FIR and IIR in FPGA, *2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, (2015), 958-963

[5] Online: https://altera.com/documentation/hco1421694595728.html

[6] Poczekajło P., Wirski R., Synthesis and Realization of 3-D Orthogonal FIR Filters Using Pipeline Structures, *Circuits Systems and Signal Processing*

[7] Poczekajło P., Wirski R.T., Synteza separowalnych trójwymiarowych filtrów ortogonalnych o strukturze potokowej, *Przegląd Elektrotechniczny*, 89, (2013), nr. 10, 150-152

[8] Poczekajło P., Wawryn K., Hardware implementation of 3D pipelined laplace filter based on rotation structures, *2017 MIXDES - 24th International Conference "Mixed Design of Integrated Circuits and Systems*, Bydgoszcz, (2017), 276-280

[9] Poczekajło P., Implementacja sprzętowa potokowego filtru uśredniającego 3D w układzie FPGA, *Przegląd Elektrotechniczny*, 93, (2017), nr. 8, 17-19

[10] Vaidyanathan P. P., Multirate Systems And Filter Banks, *Prentice Hall*, (1993)

[11] Online: https://www.terasic.com.tw/cgi-bin/page/archive.pl?CategoryNo=167&No=816

[12] Volder J.E., The CORDIC trigonometric computing technique, *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, 330–334, 1959