Mariusz Kapruziak
Bogdan Olech

Politechnika Szczecińska,
Wydział Informatyki
ul. Żołnierska 49

mkapruziak@wi.ps.pl
bolech@wi.ps.pl

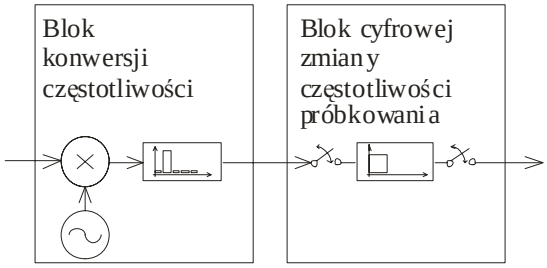
PWT 2004
Poznańskie Warsztaty Telekomunikacyjne
Poznań 9 - 10 grudnia 2004

Dedykowany procesor do cyfrowej konwersji częstotliwości w dół

***Streszczenie*: Realizacja dedykowanego procesora cyfrowej konwersji częstotliwości w dół przedstawiona w niniejszej pracy, w zasadniczym zarysie, została ujęta w świetle ogólniejszych rozważań dotyczących organizacji dedykowanego procesora radia programowalnego. W tekście, kolejno przedstawione zostały propozycje organizacji poszczególnych elementów procesora, wstęp do ogólniejszych rozważań nad organizacją procesora radia programowalnego oraz propozycja możliwego kierunku dalszego rozwoju procesorów tego typu.**

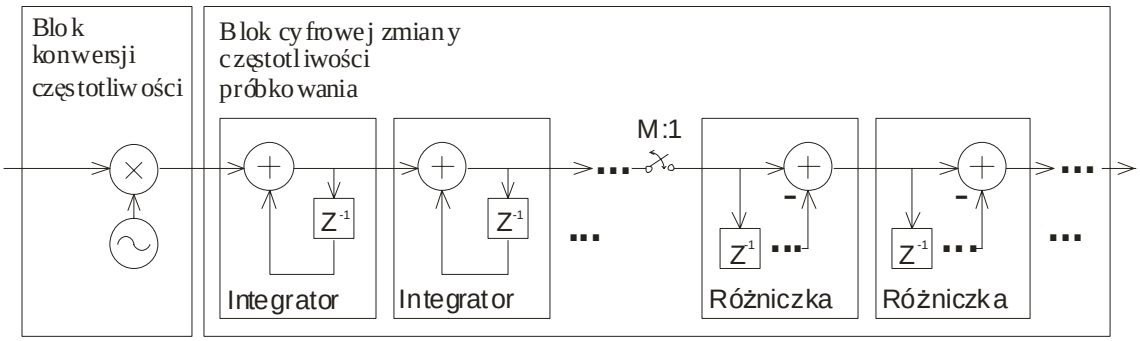
1. CYFROWA KONWERSJA CZĘSTOTLIWOŚCI

Cyfrowa konwersja częstotliwości jest połączeniem dwóch operacji: konwersji częstotliwości i cyfrowej zmiany częstotliwości próbkowania (rys.1). Blok konwersji częstotliwości składa się zazwyczaj z układu cyfrowego miksera, układu DDS (*Direct Digital Synthesis*) i filtru. Blok cyfrowej zmiany częstotliwości próbkowania to natomiast „wielorozdzielczy" filtr interpolacyjny [1], na którego wejściu podawany jest sygnał o jednej częstotliwości próbkowania, w wyniku przekazujący sygnał o częstotliwości innej.



Rys.1 Cyfrowa konwersja częstotliwości

Zaprojektowanie wspomnianego filtru interpolacyjnego, jak również uniwersalnego bloku konwersji częstotliwości w ogólności nie jest zadaniem trywialnym. Istnieje cały szereg metod które pozwalają dobrać odpowiednią architekturę jak również współczynniki filtrów i elementów [1]. W szczególnym przypadku zadanie jednak upraszcza się dość istotnie, gdy dotyczy ono wydobycia składowej o wąskim paśmie z sygnału o szerokim paśmie. Przypadek ten jest bardzo częsty w konstrukcji terminala radia programowalnego, na przykład przy cyfrowej konwersji z częstotliwości pośredniej (IF) do pasma podstawowego (BB). Możliwe jest wtedy użycie efektywnej realizacji filtru CIC (*Cascade Integrator Comb Filters*) [2], poprzedzonej bezpośrednio blokiem miksera (rys.2).

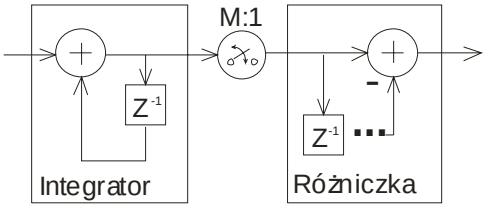


Rys.2 Cyfrowa konwersja częstotliwości z użyciem filtracji typu CIC

Cyfrową konwersję częstotliwości z użyciem fitracji CIC można przeprowadzić na bazie relatywnie prostej struktury. Możliwe jest również jej usztywnienie nie tracąc przy tym wiele z uniwersalności układu. Przykładem takiego wykonania jest seria Analog Devices AD662x [3]. Usztywnienie struktury powoduje jednak, że musi zostać zawarty pewien kompromis z uwagi na praktyczną implementację. Oznaczać to będzie zawsze zmniejszenie efektywności użycia zasobów i ograniczenie elastyczności.

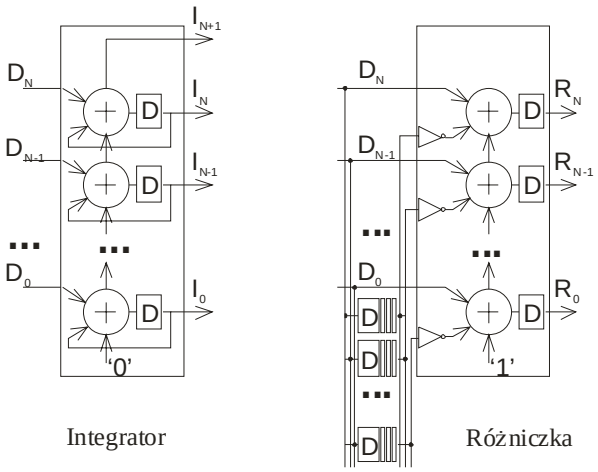
2. FILTRACJA TYPU CIC

Filtr CIC jest kaskadowym połączeniem trzech bloków: integratora, bloku zmiany częstotliwości próbkowania (*down-sampler*) i bloku różniczki (rys.3). Blok integratora jest operacją bieżącej akumulacji wartości sygnału (rys.4). Efektywna realizacja takiej operacji wymagałaby użycia bloku sumatora z możliwością natychmiastowego powrotu sygnału z wyjścia na wejście. Istnieje w tym przypadku jeszcze zagadnienie ilości bitów do reprezentacji wyniku, która będzie się zwiększać na każdym kolejnym integratorze w filtrze CIC.



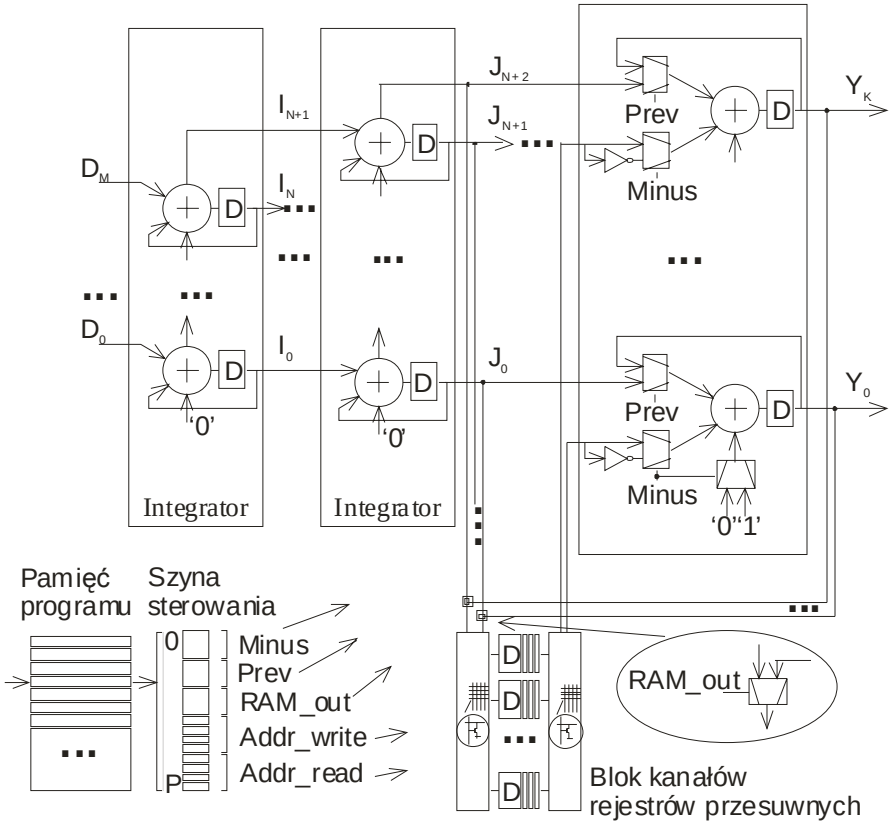
Rys.3 Filtr typu CIC

Blok różniczki wymaga natomiast użycia członów opóźniających, czyli elementów pamiętających ułożonych w strukturę kolejki. Struktura odejmowania nie wymaga tym razem aby wynik wracał na wejście (rys.4).



Rys.4 Propozycja struktury bloku integratora i różniczki

Blok zmiany częstotliwości próbkowania, nie musi być widoczny w ścieżce danych. Można go umieścić w części sterującej. Do realizacji przedstawionego zadania zaproponowano dedykowany procesor, o ścieżce danych tak jak na rys.5. Należą do niej trzy bloki sumowania, z których ostatni jest programowalny, oraz tablica rejestrów przesuwnych o konfigurowalnej długości.



Rys.5 Propozycja struktury procesora cyfrowej konwersji częstotliwości w dół

Instrukcja dla proponowanego procesora jest relatywnie niedługa i ma wielkość:

Długość_słowa_instrukcji [bity] = (Minus + Prev + RAM_out) + (adr_kanału_read + adr_kanału_write)

Dlugość_słowa_instrukcji = (3b) + (3b+3b) = 9b

Przy obliczaniu długości instrukcji założono, że adres kanału jest 3-bitowy, czyli możliwe jest zaadresowanie 8 kanałów. Jako kanał rozumiany jest skonfigurowany (o określonej długości) blok rejestrów przesuwnych (ilość rejestrów w bloku zgodna z ilością elementów sumujących w ostatnim bloku sumowania). Liczba kanałów może być określona dowolnie na etapie projektowanie procesora, natomiast później pozostaje stała. Określa ona jednocześnie możliwą różnorodność wykonywanych instrukcji, czyli maksymalną liczbę słów instrukcji w pamięci programu.

3. PRZETWARZANIE WIELOTOROWE

Aby umożliwić przetwarzanie kilku niezależnych torów w procesorze z odpowiednią, podkrotną częstotliwości próbkowania, należy wykonać kilka istotnych zmian w jego strukturze. Zmiany koncentrują się na fakcie, że każdy z bloków, a nie tylko ostatni, musi mieć dostęp do bloków z rejestrami przesuwnymi. Konieczne jest w tym przypadku stworzenie konfigurowalnej matrycy przełączeń, łączącej odpowiednie linie z bloków do odpowiednich „kanałów" bloku pamięci (rys.6).

Dość istotnie rośnie w tym przypadku długość słowa instrukcji. Dla każdego z bloków musi zostać wyspecyfikowana konfiguracja, oraz adres kanału pamięci.

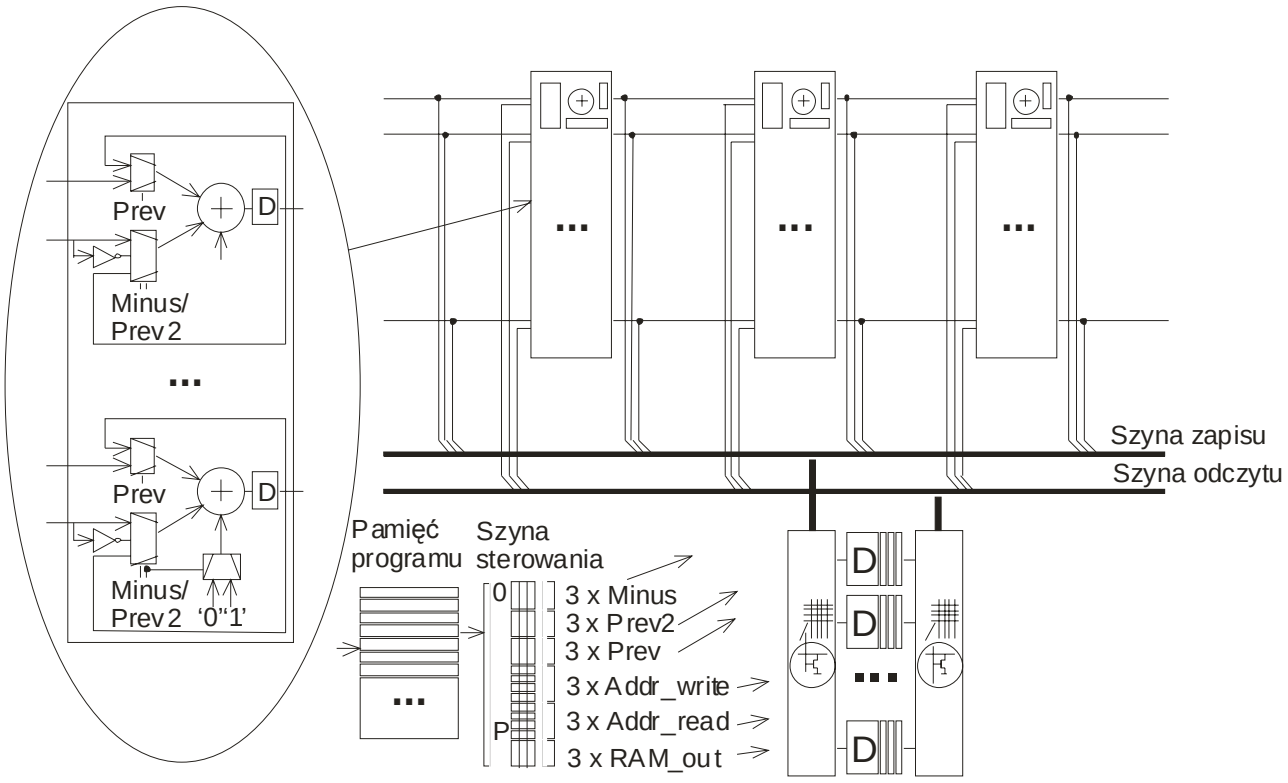
*Długość_słowa_instrukcji [bity] = ilość_bloków * ((Minus + Prev2 + Prev) + (adr_kanału_read + adr_kanału_write))*

Dlugość_słowa_instrukcji = 3 · (3b + (3b + (3b)) = 27b

Opłacalną strategią mogłoby być zatem hierarchiczne ułożenie pamięci programu. Program właściwy mógłby się składać z indeksów do osobnego bloku, przechowującej pełny kod instrukcji. Blok ten mógłby być pamięcią, bądź skonfigurowaną odpowiednio strukturą rekonfigurowaną. Wypełnienie tego bloku mogłoby zajść jednorazowo dla danego algorytmu, w czasie jego inicjalizacji. Takie podejście pozwoliłoby na zmniejszenie ilości bitów instrukcji programu, umożliwiając jednocześnie wykorzystanie elastyczności organizacji procesora.

4. MNOŻENIE

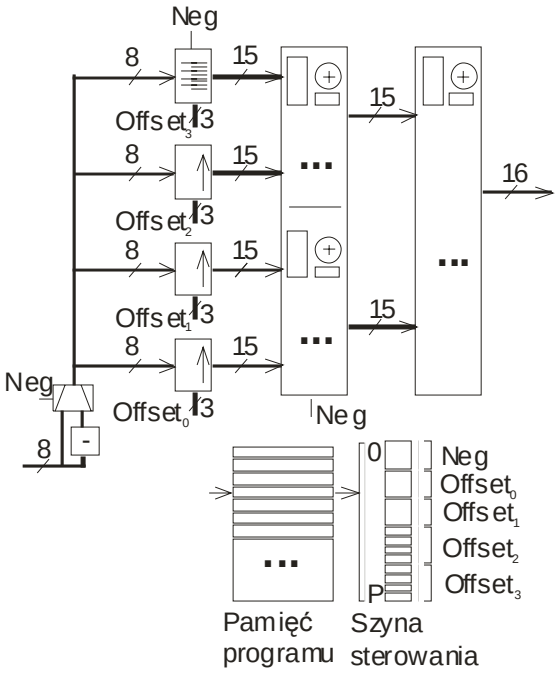
Realizacja mnożenia jest interesująca w tym przypadku z dwóch powodów. Po pierwsze operacja jest typowa i niezwykle częsta w algorytmach DSP. W przypadku proponowanego procesora występuje w bloku konwersji częstotliwości.



Rys.6 Propozycja struktury wielokanałowego procesora cyfrowej konwersji częstotliwości w dół

Po drugie natomiast realizacja mnożenia dla dowolnej stałej wymaga przynajmniej dwukrotnie mniej zasobów niż realizacja mnożenia dla dwóch zmiennych. W przypadku proponowanego procesora jedna ze zmiennych ma charakter bezpośredni i pochodzi wprost z kodu (z bloku DSS), jest znana przed wykonaniem mnożenia. W związku z tym możliwe jest jego dużo bardziej oszczędne wykonanie. Proponowana struktura byłaby zbliżona do opisywanej poprzednio z tą jednak różnicą, że wymagałaby użycia bloków przesunięcia bitowego na wejściu (rys.7). Wywodzi się ona z modyfikacji równoległej struktury układu mnożącego (*fast array multiplier*) [4].

Do mnożenia dwóch liczb 8-bitowych potrzebne są w takim wypadku trzy 15-bitowe sumowania. W zależności od jednej z liczb zmieniają się współczynniki przesunięcia bitowego na wejściach, oraz ewentualne negowanie liczby na wejściu. Struktura opiera się na tym, że liczba wartości do sumowania wynosi najwyżej 4, lub odejmowania, jeśli liczba jedynek była większa niż 4. Przy odejmowaniu maksymalną wartość, od której należy odejmować stosunkowo łatwo można określić.



Rys.7 Proponowana struktura mnożnika

Implementacja takiej struktury w rekonfigurowalnej tablicy FPGA Xilinx Spartan II nie jest w stanie jednak wykazać wspomnianych zalet. Blok powstały w wyniku takiego podejścia zużywa dwukrotnie więcej zasobów niż mnożenie wykonane przez narzędzie syntezy. Dzieje się tak, dlatego że operacja przesunięcia bitowego o zmienną wartość realizowana jest w tym przypadku jako układ logiczny, który pojedynczo zajmuje ponad 1/3 realizacji samego mnożenia. Struktura zakłada natomiast użycie 4 takich bloków. Rozważano wiele możliwości, ale w ostatecznej wersji procesora zdecydowano się na użycie dedykowanego układu mnożenia, pomimo elastyczności i przejrzystości proponowanego mnożnika.

5. WN0IOSKI I UOGÓLNIENIA ZWIĄZANE Z REALIZACJĄ PROCESORA DLA RADIA PROGRAMOWLNEGO

Obecnie nie jest jasne, jakiego typu architektura mogłaby stać się standardem w dziedzinie radia programowalnego. Z jednej strony podjęto szereg badań i prób nad realizacją tego typu systemów na procesorach ogólnego przeznaczenia (projekt MIT Virtual Radio [5]) czy typowych procesorach przetwarzania sygnałów (SpeakEasy [6]). Z drugiej strony natomiast coraz silniejsze wydają się wpływy układów rekonfigurowanych [7]. Początkowo mało obiecujące (wnioski z realizacji SpeakEasy faza II [8]), z powodu zbyt długiego czasu zmiany "kontekstu", wraz z rozwojem technologicznym stały się poważną alternatywą dla konwencjonalnych procesorów sygnałowych (wykazane przy realizacji projektu SDR–3000 [8] - moduł radiowy oparty na strukturze FPGA Xilinx Virtex II).

Zbyt duża "ziarnistość" konwencjonalnych układów rekonfigurowalnych (CPLD, FPGA) wciąż jednak stawia barierę na czas zmiany i "wielkość" programu systemów radiowych. Z drugiej strony obecnie wydaje się już jasne, że procesor dla radia programowalnego musiałby posiadać elementy rekonfigurowalności, nie tylko reprogramowalności. Próby zmniejszenia efektu kosztów "ziarnistości" zostały podjęte w projekcie Virginia Tech Chariot [8]. Konfiguracji podlegają tam nie pojedyncze bloki logiczne (CLB) ale połączenia pomiędzy gotowymi już układami przetwarzania sygnałów (ALU, MAC, itp.) i pamięcią, tworząc strukturę CCM (*Custom Computing Machine*). Procesor tego typu nie wszedł wciąż do szerszej produkcji, być może dlatego, że wciąż budowa radia programowalnego w terminalu bezprzewodowym nie jest opłacalna.

Pojedyncze zdobycze radia programowalnego powoli natomiast przebijają się do systemów bezprzewodowych, jak na przykład DDS czy DDC/DUC, które stopniowo stają się "standardem" realizacji terminali radiowych. Ewolucja tego typu wymusiła, że na ogromnym znaczeniu zyskały bardziej układy ASIC i ASSP, czyli elementy o tylko ograniczonej rekonfigurowalności do pewnej klasy zadań. Przykładem może być seria Analog Devices VersaCOMM [3].

Wiele wskazuje, w tej sytuacji, na to, że analizę organizacji dedykowanego procesora radia programowalnego warto rozpocząć jednak nie od ogólnych założeń i architektur, tak jak się to robi w przypadku "dużych" projektów z dziedziny, ale od głębszych rozważań nad kompromisami odnośnie konkretnej klasy algorytmów, efektywnie realizowanych przez daną strukturę. Metodykę pracy można w takim przypadku oprzeć o metodykę kosyntezy sprzętowo-programowej, czyniąc konstrukcję bardziej "krojoną" na miarę zadania, niż uniwersalną. Takie prace podjęte zostały na przykład przez Politechnikę Drezdeńską,

proponującą CATS *(Concept for Application Tailored Signal Processing*) do analizy procesora systemów bezprzewodowych trzeciej generacji [9].

W ramach rozwoju technologii coraz silniejszy nacisk położony zostanie chyba jednak na bardziej uniwersalne układy, które mogłyby łatwo "odpowiadać" także na wciąż pojawiające się nowe algorytmy i rozwiązania, nie koniecznie wpadające łatwo w klasę tych, realizowanych efektywnie przez struktury powstałe z "krojenia na miarę". Właśnie kompromis pomiędzy tymi koncepcjami można osiągnąć poprzez właściwy rozkład struktury do algorytmu na część strukturalną i proceduralną, czyli efektywne połączenie koncepcji reprogramowalności i rekonfigurowalności. To właśnie dynamiczny kompromis między częścią strukturalną i proceduralną wydaje się naczelnym zagadnieniem w konstrukcji dedykowanego procesora radia programowalnego.

W artykule chciałem na podstawie przykładu zarysować koncepcje i pewien punkt widzenia oraz zaprosić do dyskusji nad wymaganiami i koncepcją dedykowanego procesora radia programowalnego. Poza naczelnym zagadnieniem podziału na część proceduralną i strukturalną, który wydaje się dodatkowo podkreślany w świetle szerokiego zastosowania analizy wielorozdzielczej [10], chciałem wspomnieć o innych zagadnieniach, jak opcja "hierarchicznej" struktury pamięci programu czy dynamiczny kompromis pomiędzy częścią wykonawczą i sterującą.

Problem "hierarchicznej" organizacja pamięci, w bardzo ograniczonej formie, podkreślić chciałem w ramach wspomnienia o strukturze do przetwarzania wielotorowego. Rozszerzenie tych rozważań mogłoby pozwolić na dynamiczne kontrolowanie "ogólności" instrukcji, pozwalając z jednego punktu widzenia na płynną realizację koncepcji, jak wirtualna maszyna radiowa [11], z drugiego natomiast na ograniczenie wielkości pamięci programu przy jednoczesnym pozostawieniu stopnia "ziarnistości" rekonfiguracji na niezmienionym poziomie.

Kompromis pomiędzy częścią sterującą i wykonawczą wydaje się natomiast naturalny z racji istotnego zróżnicowania "optymalnej" struktury nie tylko od algorytmu, ale w szczególności od danych. Taką sytuację chciałem zobrazować na przykładzie mnożenia, gdzie mnożenie przez dowolną stałą wymaga co najwyżej dwa razy mniejszych zasobów niż sugerowane struktury dla mnożenia dwóch zmiennych. "Sterowanie danymi", czyli uzależnianie struktury od danych, a nie tylko algorytmu, wydaje się bardzo obiecujące, mimo że w realizacji zadania mnożenia na strukturze FPGA nie udało się tego wykazać. Bardzo często dane pochodzą nie tylko z otoczenia, ale także z pamięci programu (wynikają z algorytmu), tak jak to jest w przypadku DDS, przez co są dużo wcześniej znane, i mogą być przedstawione w programie w takiej formie jaka byłaby wygodniejsza do zmiany struktury. Zwykle na takie opracowanie, które wykonywane jest dużo rzadziej niż częstotliwość próbkowania, jest dużo więcej czasu i może być pozostawione do realizacji czysto proceduralnej. Tak jest w przypadku proponowanego mnożenia, gdzie sugeruje się zapis liczby jako szeregu przesunięć (offset) i rodzaju operacji (neg), niż wartości samej liczby.

SPIS LITERATURY

[1] T. Hentschel, *Sample Rate Conversion in Software Configurable Radios, Artech House, Norwood 2002.*

[2] E.B. Hogenauer, „An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE Trans. Acoustics. Speech Signal Proc*., vol. ASSP-29, April 1981, str. 155-162.

[3] B. Brannon, C. Cloninger, „Soft Radio Runs into Hard Standards*"*, *EE Times*, April 11, 2001.

[4] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer, Heidelberg 2001.

[5] V.G. Bose, *Design and Implementation of Software Radios Using a General Purpose Processor*, PhD Thesis, MIT, 1999.

[6] R.J. Lackey, D.W. Upmal, „Speakeasy: The Military Software Radio", *IEEE Communications Magazine*, Vol. 33, No. 5, pp. 56-61, May 1995.

[7] M. Cummings, S. Haruyama, „FPGA in the Software Radio", *IEEE Communications Magazine*, Vol. 37, No. 2, pp. 108-112, February 1999.

[8] J.H. Reed, *Software Radio: A Modern Approach to Radio Engineering*, Prentice Hall PTR, 2002.

[9] G. Fettweis, „DSP Cores for Mobile Communications: Where are we going? *"*, *Proc. of ICASSP 1997*, pp. 279-282.

[10] F.J. Harris, *Multirate Signal Processing For Communication Systems*, Prentice Hall PTR, May 2004.

[11] M. Gudaitis, J. Mitola, III, „The Radio Virtual Machine", 2000.