Sebastian PAWLAK
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Implementacja filtrów cyfrowych o skończonej odpowiedzi impulsowej (FIR) w układzie FPGA

Mgr inż. Sebastian PAWLAK

Autor artykułu ukończył studia na kierunku Elektrotechnika, specjalizacji Inżynieria Systemów Komputerowych na Uniwersytecie Zielonogórskim, gdzie pracuje obecnie na stanowisku asystenta. Przygotowuje się także do otwarcia przewodu doktorskiego. Głównym obszarem zainteresowań badawczych Autora jest wykorzystanie układów FPGA do akceleracji obliczeń w systemach przetwarzania obrazów.

e-mail: S.Pawlak@iie.uz.zgora.pl

Streszczenie

Artykuł przedstawia wyniki badań w zakresie implementacji filtrów cyfrowych o skończonej odpowiedzi impulsowej (FIR) w układzie FPGA. Wykonane zostało porównanie parametrów czasowych oraz zużycia zasobów sprzętowych wybranych architektur filtrów cyfrowych.

Słowa kluczowe: cyfrowe przetwarzanie sygnałów, filtry cyfrowe, układy logiki programowalnej, FPGA.

Implementation of digital Finite Impulse Response (FIR) filters in the FPGA device

Abstract

This paper describes implementation of a digital Finite Impulse Response.(FIR) filters in the Field Programmable Gate Array (FPGA) device. Parameters of different architectures of digital FIR filters were compared.

Keywords: digital signal processing, digital filters, programmable logic devices, FPGA.

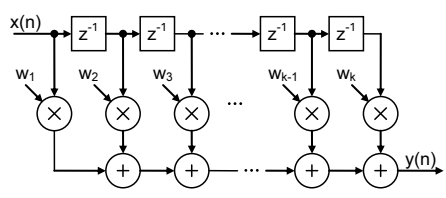
1. Wstęp

Filtry cyfrowe są jednym z głównych elementów systemów cyfrowego przetwarzania sygnałów. Ze względu na właściwości, filtry cyfrowe dzielą się na:
- dolnoprzepustowe,
- górnoprzepustowe,
- pasmowoprzepustowe,
- pasmowozaporowe.

Można wyróżnić następujące typy filtrów cyfrowych:
- filtry o skończonej odpowiedzi impulsowej (FIR),
- filtry o nieskończonej odpowiedzi impulsowej (IIR),
- filtry adaptacyjne,
- filtry nieliniowe (np. filtr medianowy).

Najprostsze w implementacji są filtry cyfrowe o skończonej odpowiedzi impulsowej, gdyż nie posiadają one sprzężenia zwrotnego.

Strukturę filtra cyfrowego FIR *k*-tego rzędu przedstawiono na rysunku 1.



Rys. 1. Struktura filtra cyfrowego o skończonej odpowiedzi impulsowej
Fig. 1. Structure of a digital Finite Impulse Response filter

Działanie filtra polega na obliczaniu splotu ciągu próbek sygnału źródłowego z wektorem wag. Działanie filtra opisuje wzór:

$$y(n) = \sum_{i=0}^{k-1} x(n-i)w_{i+1} \tag{1}$$

gdzie:

y(n) – bieżąca wartość na wyjściu filtra
x(n) – bieżąca próbka sygnału źródłowego
W – wektor współczynników określających rodzaj filtracji
$W = \{w_1, w_2, \ldots, w_{k-1}, w_k\}$

Obecnie do budowy filtrów cyfrowych wykorzystuje się najczęściej procesory sygnałowe (DSP). Posiadają one w swojej strukturze dedykowane bloki zwane buforami kołowymi (Circular Buffer) z automatyczną inkrementacją (lub dekrementacją) wskaźników danych. Dzięki temu możliwe jest zwiększenie wydajności podczas liczenia splotu, jednak podobnie jak w przypadku procesorów ogólnego przeznaczenia, także procesory sygnałowe wykonują operacje mnożenia i sumowania (MAC) sekwencyjnie. Im wyższy rząd filtra, tym więcej sekwencyjnych operacji przypada na jedną próbkę sygnału wyjściowego.

2. Zastosowanie układów FPGA do budowy filtrów cyfrowych

Układy FPGA (Field Programmable Gate Array) składają się przede wszystkim z wielu identycznych bloków logicznych zwanych plastrami (Slice). Można z nich budować prawie dowolny system cyfrowy, którego architekturę określa programista. Podstawowymi zaletami układów FPGA w porównaniu do procesorów sygnałowych są:
- możliwość zrównoleglenia operacji,
- możliwość budowy systemów cyfrowych o dowolnej precyzji (szerokości słowa danych),
- możliwość reprogramowalności (nowoczesne układy pozwalają podmieniać pewne bloki w trakcie pracy układu).

Zrównoleglenie operacji w odniesieniu do filtrów cyfrowych umożliwia np. jednoczesne liczenie wszystkich iloczynów splotu, co prowadzi do znacznego wzrostu wydajności, a dzięki temu możliwe jest zwiększenie częstotliwości próbkowania sygnału źródłowego.

Na rynku dostępne są procesory sygnałowe 16-bitowe i 32-bitowe. Gdy projektant filtra stwierdzi, że precyzja 22-bitowa jest wystarczająca, musi wykorzystać układ 32-bitowy aby zapewnić zakładaną wydajność. Używając układu programowalnego można zbudować system dokładnie 22-bitowy. Możliwość budowy systemu o dowolnej precyzji pozwala także implementować filtry wyższych rzędów w tej samej strukturze FPGA poprzez zmniejszenie szerokości szyny danych.

Nowoczesne układy FPGA posiadają w swojej strukturze także specjalizowane bloki takie jak pamięci dwuportowe czy sprzętowe układy mnożące. Dzięki nim, w najbardziej zaawansowanych układach takich jak np. Virtex2Pro [1], czy Virtex4 [2] firmy Xilinx, możliwe jest uzyskanie wydajności rzędu 250GMAC/s (miliardów mnożeń na sekundę).

Nowością w układach FPGA (wprowadzoną wraz z pojawieniem się układów z rodziny Virtex4) są dedykowane bloki DSP48 [3]. Składają się one ze sprzętowego układu mnożącego na wyjściu którego jest 48-bitowy sprzętowy sumator/akumulator. W porównaniu do poprzedniej rodziny układów (Virtex2Pro), dzięki zastosowaniu bloków DSP48 wyeliminowano konieczność implementacji sumatorów w blokach logicznych.

3. Projektowanie filtrów cyfrowych FIR

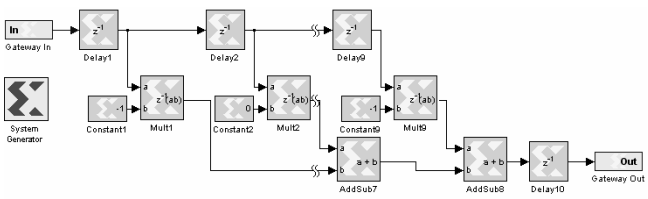
Do projektowania badanych filtrów cyfrowych wykorzystane zostało narzędzie FDATool, wchodzące w skład pakietu System Generator 8.1 firmy Xilinx. System Generator jest nakładką na środowisko MATLAB+SIMULINK R14 firmy Mathworks. Pakiet zawiera bloki, z których można budować cyfrowe systemy przetwarzania sygnałów oraz zbiór narzędzi do sprzętowej kosymulacji oraz do generowania syntezowanego kodu HDL implementującego zaprojektowane systemy.

Zaprojektowane zostały dwa filtry:
- dolnoprzepustowy filtr FIR 9-rzędu,
- górnoprzepustowy filtr FIR 27-rzędu.

Filtr dolnoprzepustowy zbudowano w ośmiu odmianach: LP9_05S, LP9_05H, LP9_08S, LP9_08H, LP9_16S, LP9_16H, LP9_24S, LP9_24H.

Poszczególne symbole oznaczają odpowiednio:
- LP9 – filtr dolnoprzepustowy (LowPass) 9-rzędu,
- 05 – 5-bitowe dane wejściowe,
- 08 – 8-bitowe dane wejściowe,
- 16 – 16-bitowe dane wejściowe,
- 24 – 24-bitowe dane wejściowe,
- S – układy mnożące i sumatory implementowane w blokach logicznych, pełna precyzja (rozszerzająca się magistrala danych uniemożliwiająca wystąpienie przekroczenia zakresu),
- H – układy mnożące implementowane jako sprzętowe dedykowane bloki mnożące (MULT18x18), sumatory implementowane w blokach logicznych, pełna precyzja.

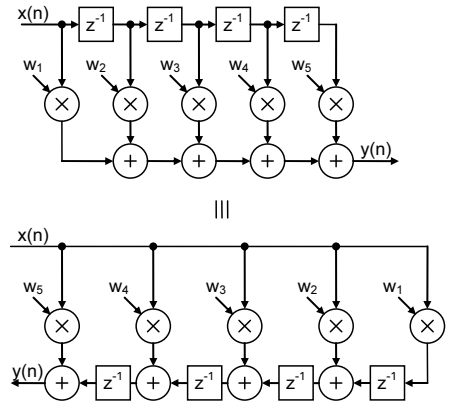
Strukturę filtra dolnoprzepustowego przedstawiono na rysunku 2.



Rys. 2. Struktura dolnoprzepustowego filtra cyfrowego 9-rzędu
Fig. 2. Structure of a 9-stage low pass digital FIR filter

Filtr górnoprzepustowy zbudowano w dwudziestu odmianach oznaczonych symbolami: HP27_16abc[_RETIMED], gdzie:
- HP27_16 – filtr górnoprzepustowy (HighPass) 27-rzędu, 16-bitowe dane wejściowe ze znakiem, kod UZ2, 1-bit części całkowitej, 14 bitów części ułamkowej,
- ab = "FP" – pełna precyzja (FullPrecision),
- a = 'T' – wszystkie sumatory 16-bitowe, obcinające bity przeniesienia z najmłodszej pozycji (Truncate),
- a = 'R' – wszystkie sumatory 16-bitowe, zaokrąglanie na najmłodszej pozycji (Round),
- b = 'W' – wszystkie sumatory 16-bitowe, „zawijanie" wyników przy przekroczeniu zakresu (Wrap),
- b = 'S' – wszystkie sumatory 16-bitowe, nasycenie przy przekroczeniu zakresu (Saturate),
- c = 'H' – układy mnożące implementowane jako sprzętowe dedykowane bloki mnożące (MULT18x18), sumatory implementowane w blokach logicznych,
- c = 'S' – układy mnożące i sumatory implementowane w blokach logicznych,
- _RETIMED – wersja po przeniesieniu bloków opóźniających za bloki mnożące.

W celu polepszenia parametrów czasowych badanego filtra górnoprzepustowano zastosowano tzw. „Retiming". Jest to technika polegająca na przeniesieniu bloków opóźniających. Technikę tą można jednak zastosować jedynie wtedy, gdy filtr ma symetryczny rozkład wag w wektorze współczynników oraz gdy nie ma sprzężenia zwrotnego. Technikę tą ilustruje rysunek 3. Zastosowanie jej pozwala na skrócenie ścieżki krytycznej filtra, a co za tym idzie, zwiększenie wydajności.



Rys. 3. Modyfikacja struktury filtra
Fig. 3. Retiming of digital FIR filter

4. Implementacja

Do syntezy i implementacji projektów wykorzystano oprogramowanie Xilinx Foundation ISE 7.1. Platformę docelową stanowił układ FPGA Virtex2Pro XC2VP30-7FF896C.

Parametry czasowe zaimplementowanych filtrów dolnoprzepustowych przedstawiono w tabeli 1.

Tab. 1. Parametry czasowe filtrów dolnoprzepustowych
Tab. 1. Timing parameters of implemented low pass filters

Filtr	t_{min} [ns]	f_{max} [MHz]
LP9_05H	15,101	66,221
LP9_05S	12,769	78,315
LP9_08H	17,598	56,825
LP9_08S	19,414	51,509
LP9_16H	21,049	47,508
LP9_16S	23,434	42,673
LP9_24H	23,712	42,173
LP9_24S	24,809	40,308

Zasoby zużyte do implementacji różnych odmian testowego filtra dolnoprzepustowego zestawiono w tabeli 2.

Tab. 2. Zużycie zasobów sprzętowych przez filtry dolnoprzepustowe
Tab. 2. Hardware resources consumption by low pass filters

Filtr	Zasoby				
	FF	LUT	Slices	MULT18x18	IOB
LP9_05H	40	116	84	9	24
LP9_05S	454	331	327	0	24
LP9_08H	64	164	116	9	33
LP9_08S	793	716	503	0	33
LP9_16H	128	292	212	9	57
LP9_16S	2621	2563	1616	0	57
LP9_24H	1722	1923	1253	36	81
LP9_24S	6132	5435	3494	0	81

Jak widać użycie sprzętowych układów mnożących znacznie ogranicza ilość zajmowanych zasobów. Należy jednak pamiętać, że nie w każdym układzie FPGA dostępna jest znaczna ilość dedykowanych układów mnożących. W zależności od wersji układu z rodziny Virtex2Pro, dostępnych jest od 12 do 444 bloków MULT18x18. W testowanym XC2VP30 jest ich 136.

W przypadku filtra LP9_24H zużycie sprzętowych mnożarek jest trzykrotnie wyższe niż rząd filtra. Wynika to z faktu wykorzystania 24-bitowej magistrali danych wejściowych. Bloki

MULT18x18 mają dwa wejścia 18-bitowe – dlatego, aby pomnożyć dwie 24-bitowe liczby potrzebne są aż trzy bloki.

Parametry czasowe zaimplementowanych filtrów górnoprzepustowych zestawiono w tabeli 3.

Tab. 3. Parametry czasowe filtrów górnoprzepustowych
Tab. 3. Timing parameters of implemented high pass filters

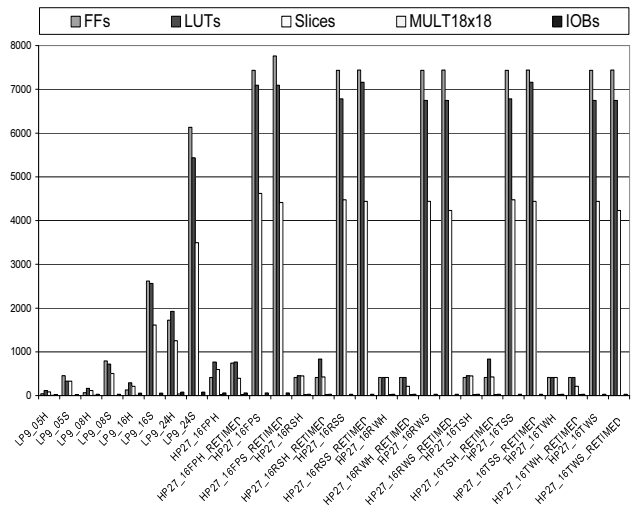
Filtr	t_{min} [ns]	f_{max} [MHz]
HP27_16FPH	50,478	19,811
HP27_16FPH_RETIMED	9,517	105,075
HP27_16FPS	55,013	18,178
HP27_16FPS_RETIMED	12,262	81,553
HP27_16RSH	91,220	10,963
HP27_16RSH_RETIMED	9,136	109,457
HP27_16RSS	95,784	10,440
HP27_16RSS_RETIMED	13,351	74,901
HP27_16RWH	43,000	23,256
HP27_16RWH_RETIMED	7,132	140,213
HP27_16RWS	48,309	20,700
HP27_16RWS_RETIMED	13,024	76,781
HP27_16TSH	87,603	11,415
HP27_16TSH_RETIMED	7,972	125,439
HP27_16TSS	91,985	10,871
HP27_16TSS_RETIMED	12,207	81,920
HP27_16TWH	43,292	23,099
HP27_16TWH_RETIMED	7,132	140,213
HP27_16TWS	48,311	20,699
HP27_16TWS_RETIMED	13,024	76,781

Widoczna jest znaczna przewaga filtrów w wersji po przekształceniu (RETIMED). Przyrosty wydajności sięgają nawet 898% dla wersji HP27_16RSH_RETIMED w stosunku do HP27_16RSH. Widoczna jest także przewaga filtrów wykorzystujących sprzętowe bloki mnożące, jednak w tym wypadku przyrost wydajności sięga jedynie 82%. Zużycie zasobów przez zaimplementowane filtry górnoprzepustowe przedstawiono w tabeli 4.

Tab. 4. Zużycie zasobów sprzętowych przez filtry górnoprzepustowe
Tab. 4. Hardware resources consumption by high pass filters

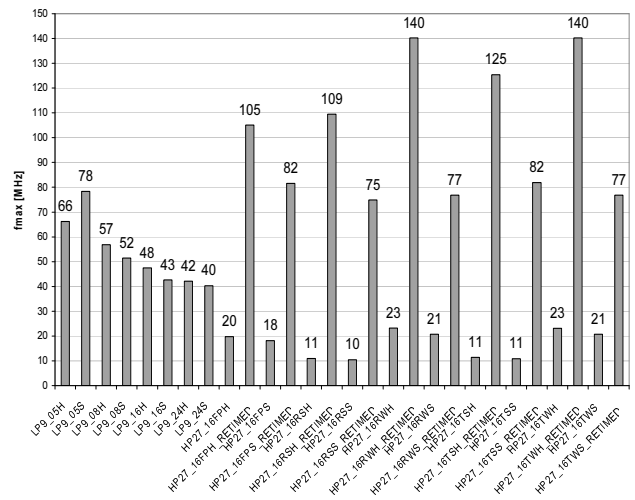
Filtr	Zasoby				
	FF	LUT	Slices	MULT18x18	IOB
HP27_16FPH	416	767	598	27	59
HP27_16FPH_RETIMED	741	767	398	27	59
HP27_16FPS	7436	7095	4621	0	59
HP27_16FPS_RETIMED	7762	7095	4414	0	59
HP27_16RSH	416	457	450	27	33
HP27_16RSH_RETIMED	416	832	425	27	33
HP27_16RSS	7436	6785	4473	0	33
HP27_16RSS_RETIMED	7437	7160	4441	0	33
HP27_16RWH	416	416	416	27	33
HP27_16RWH_RETIMED	416	416	216	27	33
HP27_16RWS	7436	6744	4439	0	33
HP27_16RWS_RETIMED	7437	6744	4232	0	33
HP27_16TSH	416	457	450	27	33
HP27_16TSH_RETIMED	416	832	425	27	33
HP27_16TSS	7436	6785	4473	0	33
HP27_16TSS_RETIMED	7437	7160	4441	0	33
HP27_16TWH	416	416	416	27	33
HP27_16TWH_RETIMED	416	416	216	27	33
HP27_16TWS	7436	6744	4439	0	33
HP27_16TWS_RETIMED	7437	6744	4232	0	33

Porównanie zużycia zasobów testowych filtrów przedstawiono na rysunku 4.



Rys. 4. Wykres zużycia zasobów sprzętowych
Fig. 4. Hardware resources consumption chart

Na rysunku 5 przedstawiono porównanie wydajności filtrów testowych.



Rys. 5. Wydajność zaimplementowanych filtrów
Fig. 5. Performance of implemented digital filters

5. Podsumowanie

Układy FPGA umożliwiają budowanie bardzo wydajnych i rozbudowanych filtrów cyfrowych. Dzięki ich elastyczności, możliwe jest szybkie wprowadzanie zmian w istniejących implementacjach.

W ciągu najbliższych kilku lat należy się spodziewać powstawania jeszcze wydajniejszych układów, a dzięki temu powinny powstać nowe, interesujące aplikacje, jak np. programowe radio (Software Radio). Ideą takiego rozwiązania jest próbkowanie bezpośrednio sygnału radiowego i jego w pełni cyfrowa obróbka (z pominięciem filtrów analogowych).

6. Literatura

[1] Xilinx: Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Datasheet, www.xilinx.com, October 2005.
[2] Xilinx: Virtex-4 Family overview, www.xilinx.com, February 2006.
[3] Xilinx: Xtreme DSP for Virtex-4 FPGAs User Guide, www.xilinx.com, December 2005.
[4] Stranneby D.: Cyfrowe przetwarzanie sygnałów, BTC 2004.

Artykuł recenzowany