II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

SYNTEZA LOGICZNA W PROJEKTOWANIU UKŁADÓW CYFROWEGO PRZETWARZANIA SYGNAŁÓW

Tadeusz Łuba, Mariusz Rawski, Paweł Tomaszewicz

Instytut Telekomunikacji Politechniki Warszawskiej
Nowowiejska 15/19, 00-665 Warszawa

e-mail: luba@tele.pw.edu.pl

STRESZCZENIE

Celem referatu jest promocja metod i narzędzi syntezy logicznej w aspekcie ich zastosowania w projektowaniu układów cyfrowego przetwarzania sygnałów. Omawiane są aktualnie badane metody dekompozycji funkcjonalnej, w szczególności zrównoważonej, które – jak wskazują wyniki eksperymentów technologicznych i komputerowych – prawdopodobnie zdominują implementacje układów cyfrowych w strukturach FPGA. Podano wiele przykładów potwierdzających skuteczność metod dekompozycji w odwzorowaniu technologicznym układów cyfrowych typowych dla zastosowań w technice DSP.

1. WSTĘP

W dzisiejszych czasach trudno sobie wyobrazić dziedzinę techniki, w której cyfrowe przetwarzanie sygnałów **DSP** (*Digital Signal Processing*) nie miałoby zasadniczego znaczenia. Odnosi się to do wielu zastosowań w telekomunikacji, elektronice, a nawet medycynie: od urządzeń przetwarzających obrazy, dźwięki, dane alfa-numeryczne na sygnały (i odwrotnie), przez sieci telekomunikacyjne przenoszące i komutujące te sygnały, aż po systemy urządzeń, zapewniające dostarczanie różnorakich treści, zwanych powszechnie usługami. Można tu wymienić takie przykłady, jak: modemy telefoniczne, systemy cyfrowego szerokopasmowego dostępu abonenckiego **DSL** (*Digital Subscriber Line*), dostępu radiowego, rozpoznawanie mowy i mówców, telewizja cyfrowa wysokiej rozdzielczości, telefonia komórkowa i satelitarna, czy związana z większością wymienionych systemów – kryptografia. Trzeba podkreślić, że sprzętowe realizacje układów DSP są ostatnio doskonale wspomagane przez cyfrowe technologie oparte na układach reprogramowalnych. Obecnie możliwa już jest realizacja zaawansowanych algorytmów DSP w pojedynczych strukturach FPGA [7].

Jednocześnie ze względu na ograniczoną liczbę dostępnych bloków logicznych w strukturach FPGA ogromnego znaczenia nabierają narzędzia syntezy logicznej, których zadaniem jest

optymalne umieszczenie (odwzorowanie) tworzonej aplikacji w docelowym układzie scalonym.

Wpływ zaawansowanych procedur syntezy logicznej na jakość implementacji sprzętowych układów przetwarzania informacji i sygnałów jest szczególnie znaczący w algorytmach wykorzystujących struktury FPGA typu LUT (*Look UP Table*). Bezpośrednią przyczyną tej sytuacji jest niedoskonałość procedur odwzorowania technologicznego tradycyjnie przystosowanych do struktur typu bramkowego, dla których podstawową metodą optymalizacji jest minimalizacja i faktoryzacja funkcji boolowskich. Transformują one sumo-iloczynowe wyrażenia boolowskie na wielopoziomowe postacie wyrażeń silnie sfaktoryzowanych, a te dopiero odwzorowywane są na komórki typu LUT. Jest to postępowanie sprzeczne z naturą komórki o tyle, że z punktu widzenia syntezy logicznej jest ona przystosowana do realizacji dowolnej funkcji logicznej o argumentach wprowadzonych na jej wejścia. Z tych powodów dla struktur FPGA znacznie skuteczniejszą metodą syntezy okazuje się dekompozycja funkcji boolowskich, której istotą jest synteza funkcji boolowskich w strukturach wielopoziomowych złożonych z komponentów w postaci bloków logicznych typu LUT specyfikowanych pierwotnymi tablicami prawdy. Skuteczność dekompozycji funkcjonalnej została potwierdzona w wielu pracach o charakterze teoretycznym [1], [2], [4], [9], [11], [12], [13]. Stosunkowo mało jest prac, w których procedury dekompozycji funkcjonalnej byłyby konfrontowane z analogicznymi narzędziami syntezy stosowanymi w komercyjnych systemach projektowania. Bezpośrednią przyczyną takiej sytuacji jest brak odpowiednich interfejsów przekształcających strukturę uzyskaną na zewnątrz systemu w strukturę specyfikowaną zgodnie z zasadami systemu komercyjnego. Drugą przyczyną jest złożoność obliczeniowa procedur dekompozycji funkcjonalnej, a w konsekwencji trudność realizacji procedur automatycznych. Trudności te – przynajmniej częściowo – zostały zlikwidowane w metodzie tzw. dekompozycji zrównoważonej [5], [8].

Metoda ta polega na iteracyjnym stosowaniu różnych strategii dekompozycji – stosowana jest dekompozycja szeregowa rozłączna i nierozłączna oraz równoległa. Warunek wystarczający dekompozycji szeregowej jest sformułowany w modelach algebry ternarnej i rachunku nakryć [1], a jego istotą jest odpowiedni algorytm weryfikacji dekompozycji. Cechą charakterystyczną metody jest wpływ odpowiedniej równowagi między dekompozycją szeregową a równoległą na liczbę komórek i liczbę poziomów układu zdekompowanego. Metoda dekompozycji zrównoważonej została zaimplementowana w systemie DEMAIN opracowanym w Zakładzie Podstaw Telekomunikacji na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej.

2. SYSTEM DEMAIN

System DEMAIN[1)] jest pakietem oprogramowania do dekompozycji i odwzorowania technologicznego złożonych układów logicznych opisywanych tablicami zapisanymi w standardzie berkeley'owskim z przeznaczeniem ich do realizacji w strukturach programowalnych typu FPGA. Jest on również wyposażony w programy translujące pozwalające na automatyczną konwersję sieci bloków opisanej w odpowiednich plikach wyjściowych na pliki w formacie BLIF (*Berkeley Logic Interchange Format*) oraz AHDL (*Altera Hardware Description Language*). Umożliwia to wykorzystanie programu DEMAIN jako narzędzia wspomagającego syntezę logiczną w niektórych systemach komercyjnych jak na przykład w systemie MAXplus+II lub Quartus firmy ALTERA.

Algorytm wykorzystany w programie DEMAIN jest oparty na oryginalnej metodzie łączącej dwie procedury: dekompozycję szeregową oraz dekompozycję równoległą. Procedury te są dokładnie omówione w [5]. Tu przypomnimy tylko najważniejsze zasady ich działania w celu dokładniejszego omówienia programu DEMAIN.

Dekompozycja szeregowa polega na podziale zmiennych wejściowych funkcji F na dwa rozłączne zbiory U i V tak, aby wejścia należące do podzbioru V mogły być wejściami pewnego bloku G o zadanej liczbie wejść i wyjść (gdzie liczba wyjść z bloku G jest mniejsza niż liczba wejść do tego bloku). W związku z powyższym wejścia należące do zbioru U i wyjścia bloku G tworzą razem zbiór wejść bloku H, co jest zapisywane standardowa formułą, $F = H(U, G(V))$.

Drugą procedurą stosowaną w programie DEMAIN jest dekompozycja równoległa. Polega ona na podziale zbioru funkcji wyjściowych na dwa rozłączne podzbiory i realizacji każdego z nich oddzielnie. W przypadku, gdy dekompozycja równoległa jest obliczana na poziomie pierwotnych zależności funkcjonalnych, jej wpływ na jakość struktury wielopoziomowej może być bardzo istotny. Zastosowanie dekompozycji równoległej powoduje rozkład danej funkcji na dwie funkcje o mniejszej złożoności, co ułatwia znalezienie dobrych dekompozycji szeregowych dla każdej z nich.

3. FILTRY CYFROWE

Filtry cyfrowe wykorzystuje się do zmiany atrybutów sygnału zarówno w dziedzinie czasowej jak i częstotliwościowej poprzez proces zwany splotem liniowym [7]. Proces ten w formalny sposób opisuje poniższa formuła:

$$y[n] = x[n] * f[n] = \sum_k x[k] \cdot f[n-k] = \sum_k x[k] \cdot c[k],$$

[1)] System DEMAIN jest dostępny nieodpłatnie na stronie internetowej Zakładu Podstaw Telekomunikacji ITPW: http://www.zpt.tele.pw.edu.pl

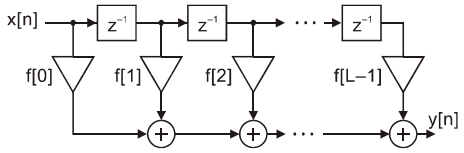
gdzie wartości $c[i] \neq 0$ nazywane są współczynnikami filtru.

Filtry cyfrowe dzieli się na filtry o skończonej odpowiedzi impulsowej FIR (*Finite Impulse Response*) i filtry o nieskończonej odpowiedzi impulsowej IIR (*Infinite Impulse Response*). Zgodnie z nazwą filtry FIR do obliczenia pojedynczej próbki sygnału wyjściowego wymagają skończonej liczby próbek sygnału wejściowego, co redukuje równanie splotu liniowego do sumy skończonej liczby próbek wejściowych. W przypadku filtru IIR niezbędne jest obliczenie nieskończonej sumy. W artykule opisano metody realizacji filtrów o skończonej odpowiedzi impulsowej ze stałymi współczynnikami LTI (*Linear Time-Invariant Filters*) FIR.

Odpowiedź $y[n]$ filtru FIR rzędu L na skończoną liczbę próbek wejściowych $x[n]$, opisana jest przez skończoną wersję sumy splotu liniowego:

$$y[n] = \sum_{k=0}^{L} x[k] \cdot c[k]$$

Filtr LTI typu FIR rzędu L przedstawiono schematycznie na rysunku 1. Zbudowany jest on z linii opóźniającej, sumatorów oraz układów mnożących.



Rys. 1. Bezpośrednia realizacja filtru FIR.

Dzięki zastosowaniu odpowiedniego oprogramowania można bardzo łatwo obliczyć współczynniki projektowanego filtru. Jednakże najtrudniej jest dokonać efektywnego odwzorowania struktury filtru FIR w architekturę docelową. Zazwyczaj filtry realizuje się jako algorytm MAC (*Multiply And Accumulate*) z wykorzystaniem wyspecjalizowanych układów DSP. W przypadku realizacji filtrów w strukturach programowalnych dla osiągnięcia dużych szybkości działania i niewielkiego wykorzystania zasobów układu programowalnego preferuje się realizacje w formie bezpośredniej lub transponowanej. Zwiększenie efektywności tego typu realizacji filtrów w strukturach programowalnych możliwe jest przez optymalizację realizacji układów mnożących i sumatorów wykorzystanych do implementacji.

Całkowicie inną architekturę filtru można otrzymać wykorzystując koncepcję arytmetyki rozproszonej DA (*distributed arithmetic*). W odróżnieniu od tradycyjnej architektury opartej na sumowaniu wyników iloczynów (*sum-of-product*), wykorzystanie koncepcji DA umożliwia obliczenie w jednym kroku sumy iloczynów konkretnego bitu próbki wejściowej przez wszystkie współczynniki.

4. ARYTMETYKA ROZPROSZONA

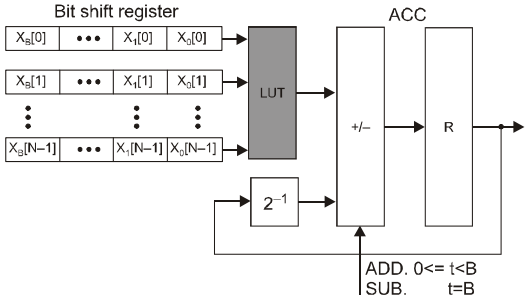
Arytmetyka rozproszona jest metodą obliczania sumy iloczynów. W wielu zastosowaniach DSP do obliczenia sumy iloczynów nie jest wymagane zastosowanie uniwersalnego układu mnożącego. W przypadku realizacji filtrów, których współczynniki są stałe w czasie, mnożenie $x[n]c[n]$ staje się mnożeniem przez stałą. Jeśli w takim przypadku weźmie się pod uwagę fakt, że wartość $x[n]$ jest liczbą binarną:

$$x[n] = \sum_{b=0}^{B-1} x_b[n] \cdot 2^b, \text{ gdzie } x_b[n] \in [0,1],$$

to do obliczenia splotu liniowego można użyć formuły zaprezentowanej poniżej.

$$y[n] = \sum_{b=0}^{B-1} 2^b \cdot \sum_{k=0}^{L} x_b[k] \cdot c[k] = \sum_{b=0}^{B-1} 2^b \cdot \sum_{k=0}^{L} f(x_b[k], c[k]) \cdot$$

Efektywność realizacji filtru opartej na tej koncepcji w znacznej mierze zależy od sposobu realizacji funkcji $f(x_b[k],c[k])$. Najczęściej funkcję tę realizuję się jako moduł kombinacyjny o L wejściach. Schematyczną reprezentację takiej realizacji przedstawiono na rysunku 2, gdzie odwzorowanie f jest reprezentowane przez tablicę LUT zawierającą wszystkie możliwe kombinacje liniowe wartości współczynników filtru i wartości bitów próbek wejściowych. Odpowiednie oprogramowanie, służące do obliczania zawartości tablicy LUT dla zadanych współczynników filtru można zaleźć w literaturze [7].



Rys. 2. Architektura DA z wykorzystaniem tablicy lookup (LUT)

Dla potrzeb eksperymentów przedstawionych w tym artykule wykorzystano odmianę architektury DA, która umożliwia zwiększenie szybkości działania filtru kosztem dodatkowych bloków LUT, rejestrów i sumatorów. Podstawowa architektura DA dla obliczenia sumy L iloczynów w jednym kroku wykorzystuje po jednym bicie każdego z L słów wejściowych. Szybkość obliczeń można zwiększyć dokonując w pojedynczym kroku obliczeń dla większej liczby bitów słów wejściowych. Maksymalną szybkość obliczeń osiąga się przez zastosowania w pełni potokowej zrównoleglonej architektury DA. Taka realizacja przewyższa szybkością obliczeń wszystkie komercyjne programowalne procesory sygnałowe.

Dla zadanych współczynników $c[i]$ filtru można bardzo łatwo opisać tablicę LUT wykorzystując język HDL. Z tego jednak względu, że rozmiar tablicy LUT rośnie wykładniczo wraz ze wzrostem liczby wejść, jakość implementacji tej tablicy ma niezwykle istotny wpływ na ostateczną wielkość zasobów wykorzystanych przy realizacji całego układu. W metodzie zaprezentowanej w tym artykule do celów odwzorowania bloku DA w komórki logiczne struktury FPGA skutecznie wykorzystano dekompozycję zrównoważoną. Dla przykładu blok DA filtru FIR F5 [3] ze współczynnikami [3, 0, –25, 0, 150, 256, 150, 0, –25, 0, 3] można opisać tablicą o 11 wejściach i 11 wyjściach. Po dokonaniu dekompozycji z wykorzystaniem oprogramowania DEMAIN, blok ten można zrealizować przy wykorzystaniu 32 komórek logicznych. Realizacja tego samego bloku dokonana bezpośrednio przez systemu QuartusII v4.0 SP1 wymaga 45 komórek logicznych. W konsekwencji realizacja całego filtru oparta na architekturze zrównoleglonej w sytuacji, gdy zastosowana zostanie dekompozycja zajmuje 530 komórek logicznych zamiast 642 komórek.

5. WYNIKI EKSPERYMENTALNE

Poniżej przedstawiono wyniki zastosowania dekompozycji funkcjonalnej w realizacji filtrów FIR wykorzystując koncepcję arytmetyki rozproszonej. Dla celów eksperymentów wykorzystano kilka przykładów filtrów różnego rzędu. Wszystkie filtry operują na 8 bitowych próbkach wejściowych, zaś ich współczynniki można znaleźć w [3].

Dla porównania wszystkie filtry zostały zrealizowane w strukturze FLEX10K10 z wykorzystaniem systemu QuartusII v4.0 SP1. W jednej metodzie realizacji nie wykorzystano dekompozycji. W drugiej metodzie do optymalizacji logiki DA zastosowano oprogramowanie DEMAIN, zaś cała struktura filtru zrealizowana została z wykorzystaniem systemu Quartus.

Wyniki zaprezentowane w tablicy 1 wskazują na znaczący wpływ dekompozycji na jakość realizacji tablic DA. Zastosowanie zaawansowanej metody syntezy logicznej pozwoliło na ponad 40% redukcję wykorzystania zasobów.

Tab. 1. Porównanie jakości realizacji logiki DA

FIR	tablica DA wejścia/wyjścia	Liczba komórek logicznych DA	
		bez dekompozycji	z dekompozycją
F4	7 / 8	12	9
F5	11 / 11	45	32
F6	11 / 11	66	45
F7	11 / 12	48	35
F8	15 / 12	136	64
F9	19 / 16	279	159
Σ		586	344
%		100	58,7

W tablicy 2 zaprezentowano porównanie jakości realizacji całej struktury filtru. Można zauważyć, że zastosowanie dekompozycji doprowadziło do zmniejszenia liczby komórek logicznych niezbędnych do realizacji filtrów bez zmniejszenia szybkości działania tych filtrów a nawet ją zwiększając. Dla przykładu do realizacji filtru F9, który jest filtrem 19 rzędu, wymaganych jest 2624 komórek logicznych, jeśli zastosowany zostanie jedynie system Quartus. Szybkość działania tak zrealizowanego filtru charakteryzowana przez maksymalną częstotliwości taktowania wynosi f_{max} = 31,25 MHz. Zastosowanie oprogramowania DEMAIN umożliwia realizację tego filtru z wykorzystaniem jedynie 1672 komórek logicznych. Jednocześnie maksymalną częstotliwości taktowania uległa zwiększeniu i wynosi f_{max} = 38,61 MHz

Tab. 2. Porównanie realizacji filtrów

FIR	bez dekompozycji		z dekompozycją		Drzewo sumatorów
	LCs (FFs)	f_{max} [MHz]	LCs (FFs)	f_{max} [MHz]	
F4	288(194)	51,28	264(194)	53,76	potok.
	224(71)	25,25	204(71)	26,60	komb.
F5	642(271)	47,62	530(271)	52,36	potok.
	527(106)	21,05	444(106)	22,03	komb.
F6	808(274)	38,61	632(274)	41,49	potok.
	710(106)	19,27	551(106)	19,61	komb.
F7	680(286)	47,62	560(286)	47,85	potok.
	576(107)	20,62	476(107)	20,96	komb.
F8	1404(323)	42,74	834(323)	48,31	potok.
	1314(140)	22,73	746(140)	22,73	komb.
F9	2624(410)	31,25	1672(410)	38,61	potok.
	2533(175)	14,47	1564(175)	20,92	komb.
Σ	12330	382,51	8477	415,23	
%	100	100	68,8	108,6	

6. WNIOSKI

Rezultaty zaprezentowane w tym artykule wskazują, że wpływ dekompozycji na jakość realizacji typowych układów DSP – filtrów cyfrowych – staje się szczególnie istotna w sytuacji, gdy układ składa się ze złożonych bloków kombinacyjnych. Jest to typowa sytuacja w przypadku realizacji filtrów z wykorzystaniem koncepcji arytmetyki rozproszonej.

W metodzie realizacji filtrów cyfrowych zaprezentowanej w tym artykule wykorzystano zaawansowaną metodę syntezy logicznej opartej na dekompozycji funkcjonalnej. Pomimo, że zarówno koncepcja arytmetyki rozproszonej jak i dekompozycji funkcjonalnej są znane i stosowane, to nigdy nie zostały jednocześnie wykorzystane w odwzorowaniu technologicznym na struktury FPGA typu LUT W artykule przedstawiono możliwość zastosowania dekompozycji zrównoważonej zarówno do optymalizacji wykorzystania zasobów sprzętowych, jak i do optymalizacji szybkości działania układu.

LITERATURA

[1] J. Brzozowski, T. Łuba: *Decomposition of Boolean Functions Specified by Cubes*. Journal of Multiple-Valued Logic and Soft Computing . Vol. 9, pp. 377-417. Old City Publishing, Inc., Philadelphia 2003

[2] S. C. Chang, M. Marek-Sadowska, T. T. Hwang: *Technology Mapping for TLU FPGAs Based on Decomposition of Binary Decision Diagrams*, IEEE Trans. on CAD, vol.15, No. 10, pp. 1226-1236, October, 1996

[3] D. J. Goodman, M. J. Carey: *Nine Digital Filters for Decimation and Interpolation*, IEEE Trans. On Acoustics, Speech and Signal Processing 25(2), pp.121-126, 1997

[4] Y. T. Lai, K. R. R. Pan, M. Pedram: *OBDD-Based Functional Decomposition: Algorithm and Implementation*, IEEE Trans. on CAD, vol. 15, No 8, pp. 977-990, August, 1996

[5] T. Łuba(red.), M. Rawski, P. Tomaszewicz, B. Zbierzchowski: *Synteza układów cyfrowych.* WKŁ Warszawa 2003

[6] T. Łuba, H. Selvaraj: *A General Approach to Boolean Function Decomposition and its Applications in FPGA-based Synthesis*. VLSI Design, Special Issue on Decompositions in VLSI Design, vol. 3, Nos. 3-4, pp. 289-300, 1995

[7] U. Meyer-Baese: *Digital Signal Processing with Field Programmable Gate Arrays*, Second Edition, Springer Verlag, Berlin, 2004

[8] M. Nowicka: *Zrównoważona metoda odwzorowania technologicznego dla układów FPGA*. Rozprawa doktorska, Wydział Elektroniki i Technik Informacyjnych Politechniki Warszawskiej, Warszawa 1999

[9] M. Rawski, L. Jóźwiak, T. Łuba: *Functional decomposition with an efficient input support selection for sub-functions based on information relationship measures*. Journal of Systems Architecture, 47, Elsevier Science B.V., 2001

[10] M. Rawski, P. Tomaszewicz, T. Łuba: *Logic Synthesis Importance in FPGA-based Designing of Information and Signal Processing Systems*, Proceedings of International Conference on Signals and Electronic Systems ICSES'04, pp. 425-428, 13-15 September 2004, Poznań

[11] P. Sapiecha, H. Selvaraj, M. Pleban: *Decomposition of Boolean Relations and Functions in Logic Synthesis and Data Analysis*. Rough Sets and Current trends in Computing, pp. 487-494, RSCT 2000, Springer, 2001

[12] H. Selvaraj, P. Sapiecha, T. Łuba: *Functional decomposition and its applications in machine learning and neural networks.* International Journal of Computational Intelligence and Applications. World Scientific Publishing Company, Vol 1, No. 3, pp. 259-271, Imperial College Press, 2001

[13] C. Scholl: *Functional Decomposition with Application to FPGA Synthesis*. Kluwer Academic Publisher, Boston 2001