Valery SALAUYOU, Marek GRUSZEWSKI
POLITECHNIKA BIAŁOSTOCKA, WYDZIAŁ INFORMATYKI,
ul. Wiejska 45A, 15-351 Białystok

Implementacja w strukturach CPLD/FPGA komparatorów hierarchicznych z wykorzystaniem równoległo-szeregowej metody syntezy

Dr hab. inż. Valery SALAUYOU

Dr hab. inż. Valery Salauyou w latach 1980-1984 pracował jako programista w Mińsku na Białorusi. W latach 1984-2002 był pracownikiem dydaktycznym w Białoruskim Państwowym Uniwersytecie Informatyki i Radioelektroniki w Mińsku, gdzie uzyskał tytuł doktora habilitowanego nauk technicznych. Jednocześnie od 1992 roku pracuje jako adiunkt Wydziału Informatyki Politechniki Białostockiej. Zainteresowania naukowe: projektowanie systemów cyfrowych na układach programowalnych.

e-mail: valsol@mail.ru

Mgr inż. Marek GRUSZEWSKI

Ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej w 1984 r. W latach 1984–1991 zatrudniony był w PDP PAN „SONOPAN". W 1992 r. rozpoczął pracę w Instytucie Informatyki Politechniki Białostockiej. Od 2001 r. do 2013 r. kierował Laboratorium Informatyki Technicznej. Obecnie zatrudniony jest na Wydziale Informatyki PB w Laboratorium Wydziału Informatyki na stanowisku starszego specjalisty ds. informatyki. Zainteresowania naukowe: synteza układów cyfrowych.

e-mail: m.gruszewski@pb.edu.pl

Streszczenie

Praca dotyczy syntezy komparatorów binarnych w strukturach CPLD/FPGA. Do budowy komparatorów wykorzystano struktury hierarchiczne i równoległo-szeregowe metody syntezy. Badania eksperymentalne wykonano dla komparatorów 128-bitowych oraz 256-bitowych w środowisku Quartus II firmy Altera. Wybrane parametry porównano z wynikami uzyskanymi za pomocą funkcji *lpm_compare*. Dla komparatorów 128-bitowych uzyskano zmniejszenie kosztu realizacji o 13% oraz zmniejszenie ich maksymalnego czasu propagacji do 38%. W przypadku komparatorów 256-bitowych uzyskano zmniejszenie kosztu realizacji o 19% oraz zmniejszenie ich maksymalnego czasu propagacji do 54%.

Słowa kluczowe: komparator binarny, język Verilog, komparator o strukturze hierarchicznej, równoległo-szeregowa metoda syntezy, funkcja lpm_compare, struktury CPLD/FPGA.

Implementation of hierarchical comparators with the use of the parallel-serial synthesis method in CPLD/FPGA structures

Abstract

The paper deals with the problem of a binary comparator synthesis in CPLD/FPGA structures. The comparators were built with the usage of the Verilog language and the Quartus II graphics editor [10]. Section 1 describes the notion of a digital comparator, its basic usage [1-4] and research directions [6-10]. Section 2 presents the general hierarchical structure of the comparator (Fig. 1). Section 3 describes the parallel-serial method of the comparator synthesis [10]. This method was used in the first level comparator synthesis in hierarchical structures of 128-bit and 256-bit comparators. Section 4 presents the results of experimental research. The comparators were built and tested in the Altera Quartus II environment. In the experimental investigations, hierarchical comparators (128-bit and 256-bit) were compared with the comparators (128_lpm and 256_lpm) built with the direct usage of the *lpm_compare* library function of the Quartus II package. The research was conducted on two CPLD families (MAX II and MAX V) and on four FPGA families (Cyclone III, Arria II GX, Arria V GZ and Stratix III). Two parameters, the implementation cost and the maximum propagation delay, were compared. For 128-bit comparators, the implementation cost was reduced by 13% and the maximum propagation delay was reduced up to 38% (depending on the family of FPGA structures). For 256-bit comparators, the implementation cost was reduced by 19% and the maximum propagation delay was reduced up to 54% (depending on the family of FPGA structures).

Keywords: binary comparator, Verilog language, hierarchical comparator, parallel-serial synthesis method, lpm_compare function, CPLD/FPGA structures.

1. Wprowadzenie

Komparator binarny (nazywany dalej komparatorem) należy do układów kombinacyjnych i służy do porównywania wartości dwóch słów binarnych, np. A i B. Komparator tradycyjny ma trzy wyjścia i realizuje następujące funkcje: G ($A > B$), E ($A = B$), L ($A < B$), gdzie $A = a_t \ldots a_1$ oraz $B = b_t \ldots b_1$.

Przy syntezie komparatorów wystarczy zrealizować tylko dwie funkcje G i E. Funkcja L zawsze może być określona na podstawie dwóch pierwszych na podstawie zależności:

$$L = \overline{G} \cdot \overline{E}. \tag{1}$$

Komparatory należą do podstawowych komponentów systemów cyfrowych. Są kluczowymi elementami w szerokim zakresie zastosowań, tj.: w procesach obliczeniowych (grafika oraz przetwarzanie obrazów/sygnałów [1]), w układach testujących (analizatory sygnatur, wbudowane układy samotestujące [2]), w procesach poszukiwania i sortowania danych [3], jako komponenty w procesorach ogólnego przeznaczenia: pamięci asocjacyjne (skojarzeniowe), bufory TLB (*Translation Lookaside Buffer)*, bufory BTB (*Branh Target Buffer*) i wiele innych bloków porównywania argumentów w CPU [4].

Rozbudowany układ cyfrowy na ogół przedstawiany jest jako zespół standardowych i/lub oryginalnych bloków funkcjonalnych. Do takich bloków funkcjonalnych należą m.in. komparatory. Najbardziej rozpowszechnioną dzisiaj bazą elementową techniki cyfrowej są złożone programowalne układy logiczne (CPLD) oraz bezpośrednio programowalne macierze bramek (FPGA) [5].

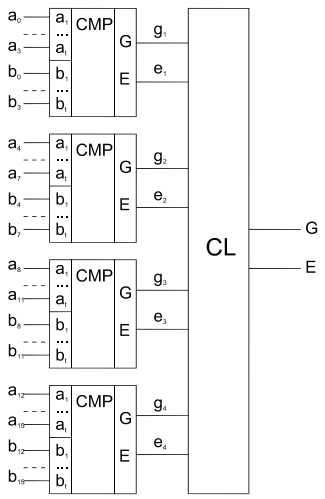
W technice obliczeniowej zauważa się stałą tendencję do zwiększania wielkości słów binarnych. Długość słów rośnie szczególnie szybko w systemach telekomunikacji, a także w urządzeniach przetwarzania i przesyłania informacji. Przy projektowaniu systemów cyfrowych pojawia się więc potrzeba opracowania efektywnych metod syntezy komparatorów w strukturach CPLD/FPGA, pracujących ze słowami binarnymi o dużych rozmiarach (komparatorów dużych rozmiarów).

W ostatnich latach zwraca się dużą uwagę na projektowanie komparatorów o dużej szybkości działania, niskim koszcie realizacji i małym poborze mocy. Przykłady wydajnych architektur komparatorów zaprezentowano w pracach [6-10]. W pracy [10] przedstawiono metody syntezy komparatorów z wykorzystaniem języka Verilog: równoległe, szeregowe, równoległo-szeregowe, a także równoległo-szeregowe z wykorzystaniem edytora graficznego. W niniejszej pracy przedstawione są wyniki syntezy hierarchicznych komparatorów w strukturach CPLD/FPGA. Wyniki syntezy porównano z biblioteczną makrofunkcją *lpm_compare* środowiska Quartus II firmy Altera.

2. Hierarchiczne struktury komparatorów

Hierarchiczna budowa komparatora dużego rozmiaru ma postać piramidalnej struktury złożonej z bloków komparatorów mniejszego rozmiaru.

Każdy blok piramidalnej struktury może zawierać z kolei inną hierarchiczną strukturę. W ten sposób można utworzyć wiele poziomów hierarchii oraz budować komparatory dowolnego rozmiaru. Na rys. 1 przedstawiono przykład komparatora 16-bitowego zbudowanego z czterech bloków 4-bitowych.



Rys. 1. Hierarchiczna struktura komparatora 16-bitowego zbudowanego z bloków 4-bitowych
Fig. 1. The hierarchical structure of a 16-bit comparator built of 4-bit blocks

Wyjściowe funkcje G oraz E generuje układ logiczny CL na podstawie następujących wyrażeń:

$$G = g_4 + e_4 \cdot g_3 + e_4 \cdot e_3 \cdot g_2 + e_4 \cdot e_3 \cdot e_2 \cdot g_1, \quad (2)$$

$$E = e_1 \cdot e_2 \cdot e_3 \cdot e_4 . \quad (3)$$

Rozbudowując wyrażenia (2) i (3) dla F bloków otrzymano uogólnione funkcje G oraz E:

$$G = g_F + e_F \cdot g_{F-1} + e_F \cdot e_{F-1} \cdot g_{F-2} + \ldots + e_F \cdot e_{F-1} \cdot \ldots \cdot e_2 \cdot g_1, \quad (4)$$

$$E = e_1 \cdot \ldots \cdot e_F. \quad (5)$$

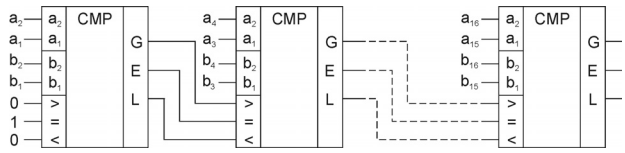
Na podstawie wyrażeń (4) i (5) zbudowano układy kombinacyjne CL dla wszystkich rozpatrywanych hierarchicznych struktur komparatorów.

3. Zastosowanie równoległo-szeregowej metody do syntezy komparatorów hierarchicznych

Równoległo-szeregowa metoda syntezy komparatorów pozwala znaleźć kompromis pomiędzy kosztem realizacji (ilością zużytych elementów logicznych) a szybkością działania (wielkością opóźnienia sygnałów I/O). Metoda ta wykazała wysoką efektywność podczas przeprowadzonych w pracy [10] badań eksperymentalnych komparatorów 64-bitowych.

Metodę równoległo-szeregową realizuje się przez szeregowe (kaskadowe) połączenie poszczególnych komparatorów mniejszego rozmiaru (sekcji). Każda sekcja, w odróżnieniu od tradycyjnego komparatora, posiada dodatkowo wejścia od poprzedniej sekcji (*gi*, *ei*, *li*), które stanowią wartości funkcji „ > , = , < ”. Każda sekcja porównuje odpowiednie części słów A i B o długości M bitów. Sekcje komparatorów połączone są szeregowo, zaczynając od najmłodszych bitów (rys. 2).

Struktury realizacji równoległo-szeregowych można ogólnie przedstawić za pomocą wyrażenia $S \times M$, gdzie: S – ilość sekcji komparatora, M – długość porównywanych słów jednej sekcji.



Rys. 2. Komparator 16-bitowy zbudowany z ośmiu 2-bitowych sekcji połączonych szeregowo
Fig. 2. The 16-bit comparator built of eight serially combined 2-bit sections

Niech A_M i B_M będą częścią słów A i B o długości M bitów. Działanie jednej sekcji komparatora można opisać za pomocą następujących równań:

$$E = (A_M = B_M) \cdot e_i, \quad (6)$$

$$G = (A_M > B_M) + (A_M = B_M) \cdot g_i, \quad (7)$$

$$L = (A_M < B_M) + (A_M = B_M) \cdot l_i, \quad (8)$$

gdzie: $(A_M = B_M)$, $(A_M > B_M)$, $(A_M < B_M)$ – wynik porównania części słów A i B.

Jeśli słowa A i B są równe dla starszych bitów ($A_M = B_M$), a także dla młodszych bitów ($e_i = 1$), to $A = B$. Jeśli $A > B$ dla starszych bitów ($A_M > B_M$), to niezależnie od wartości młodszych bitów $A > B$. Jeśli natomiast dla starszych bitów $A = B$ ($A_M = B_M$) i dla młodszych bitów $A > B$ ($g_i = 1$), to $A > B$. Analogicznie, jeśli $A < B$ dla starszych bitów ($A_M < B_M$), to niezależnie od wartości młodszych bitów $A < B$. Jeżeli dla starszych bitów $A = B$ ($A_M = B_M$) i dla młodszych bitów $A < B$ ($l_i = 1$), to $A < B$.

Do syntezy komparatorów hierarchicznych wykorzystano metodę równoległo-szeregową. Jako najmniejsze bloki pierwszego poziomu w strukturze hierarchicznej przyjęto komparatory 64-bitowe o postaci: 2x32 (M11), 4x16 (M12), 8x8 (M13), 16x4 (M14), 32x2 (M15). We wszystkich strukturach M11-M15 sekcje opisano w języku Verilog za pomocą bloku *always* [10].

Hierarchiczne struktury komparatorów 128-bitowych zbudowano z komparatorów (bloków) mniejszego rozmiaru o szerokości 64 bitów, a komparatorów 256-bitowych z bloków 64-bitowych i 128-bitowych. Do oznaczenia hierarchicznej struktury komparatora przyjęto następujące wyrażenie:

$$L\_N\text{x}T\_metoda, \quad (9)$$

gdzie: L - liczba bitów bloku drugiego stopnia, $L = N\text{x}T$; N - liczba bloków pierwszego stopnia; T - liczba bitów bloku pierwszego stopnia; *metoda* - sposób syntezy pierwszego stopnia; Podczas budowy następnego stopnia wyrażenie (9) powtarza się (tylko w miejsce wyrażenia *T_metoda* wstawia się wyrażenie bloku niższego stopnia).

Oznaczenia zastosowanych w badaniach struktur komparatorów hierarchicznych zamieszczono w opisach odpowiednich tabel z wynikami pomiarów (tabele 1 i 2).

4. Badania eksperymentalne komparatorów 128-bitowych oraz 256-bitowych

Efektywność komparatorów 128-bitowych i 256-bitowych badano za pomocą pakietu Quartus II 13.1 Web Edition (64-bit) firmy Altera. Badania przeprowadzono dla dwóch rodzin CPLD (MAX II i MAX V) oraz dla czterech rodzin FPGA (Cyclone III, Arria II GX, Arria V GZ i Stratix III). W tym celu porównano 128-bitowe komparatory hierarchiczne z komparatorem 128-bitowym (128_lpm) oraz 256-bitowe komparatory hierarchiczne z komparatorem 256-bitowym (256_lpm). Komparatory 128_lpm i 256_lpm zbudowano z bezpośrednim wykorzystaniem bibliotecznej makrofunkcji *lpm_compare* firmy Altera.

Jako kryteria porównawcze przyjęto: C - koszt realizacji mierzony ilością wykorzystanych elementów logicznych CPLD/ FPGA, D - maksymalny czas propagacji z wejścia na wyjście komparatora mierzony w nanosekundach [ns].

W tabeli 1 przedstawiono wyniki badań eksperymentalnych komparatorów 128-bitowych, a w tabeli 2 wyniki badań komparatorów 256-bitowych. Na podstawie uzyskanych wyników dokonano wyboru struktur komparatorów o najmniejszych wartościach parametrów C oraz D i porównano je z wartościami uzyskanymi bezpośrednio za pomocą funkcji *lpm_compare*.

Tab. 1. Wyniki badań eksperymentalnych komparatora 128_lpm oraz 128-bitowych komparatorów o strukturze hierarchicznej, w których do syntezy bloków pierwszego poziomu wykorzystano metodę równoległo-szeregową

Tab. 1. The results of experimental research on 128_lpm comparator and 128-bit hierarchical comparators in which the first level block synthesis was conducted with the usage of the parallel-serial method

CPLD/FPGA	Par.	Typ struktury					min_1-5	128_lpm	min_1-5/ 128_lpm
		1	2	3	4	5			
MAX II	C	218	224	231	236	235	218	213	1,02
	D	23	30	36	53	76	23	22	1,05
MAX V	C	218	224	231	236	235	218	213	1,02
	D	23	32	33	58	77	23	23	1,00
Cyclone III	C	218	228	234	224	238	218	213	1,02
	D	17	***14***	20	28	34	14	17	**0,82**
Arria II GX	C	175	168	149	158	168	149	141	1,06
	D	20	***18***	23	29	50	18	29	**0,62**
Arria V GZ	C	112	116	***101***	117	117	101	116	**0,87**
	D	25	26	32	54	46	25	21	1,19
Stratix III	C	175	168	149	158	168	158	141	1,12
	D	21	18	***17***	23	28	17	25	**0,68**

Oznaczenia typów struktur: 1) 128_2x64_M11, 2) 128_2x64_M12, 3) 128_2x64_M13, 4) 128_2x64_M14, 5) 128_2x64_M15

Tab. 2. Wyniki badań eksperymentalnych komparatora 256_lpm oraz 256-bitowych komparatorów o strukturze hierarchicznej, w których do syntezy bloków pierwszego poziomu wykorzystano metodę równoległo-szeregową

Tab. 2. The results of experimental research on 256_lpm comparator and 256-bit hierarchical comparators in which the first level block synthesis was conducted with the usage of the parallel-serial method

CPLD/ FPGA	Par.	Typ struktury										min_ 1-10	256_ lpm	min_ 1-10/ 256_lpm
		1	2	3	4	5	6	7	8	9	10			
Cyclone III	C	434	458	469	452	474	437	456	468	453	476	434	427	1,02
	D	***24***	39	48	59	67	27	26	57	48	63	24	34	**0,71**
Arria II GX	C	311	289	297	315	354	310	291	294	315	355	291	254	1,15
	D	39	35	***31***	46	54	35	35	34	40	49	31	42	**0,74**
Arria V GZ	C	200	200	***192***	228	239	202	204	194	226	232	192	236	**0,81**
	D	40	***33***	51	59	51	50	40	35	61	60	33	59	**0,56**
Stratix III	C	311	289	297	315	354	310	291	294	315	355	289	254	1,14
	D	***23***	25	24	33	40	29	25	***23***	28	36	23	50	**0,46**

Oznaczenia typów struktur: 1) 256_4x64_M11, 2) 256_4x64_M12, 3) 256_4x64_M13, 4) 256_4x64_M14, 5) 256_4x64_M15, 6) 256_2x128_2x64_M11, 7) 256_2x128_2x64_M12, 8) 256_2x128_2x64_M13, 9) 256_2x128_2x64_M14, 10) 256_2x128_2x64_M15

W grupie komparatorów 128-bitowych uzyskano zmniejszenie kosztu realizacji C o 13% (Arria V GZ, struktura 128_2x64_M13) oraz zmniejszenie maksymalnego czasu propagacji D: o 18% (Cyclone III, struktura 128_2x64_M12), o 38% (Arria II GX, struktura 128_2x64_M12), o 32% (Stratix III, struktura 128_2x64_M13).

W grupie komparatorów 256-bitowych uzyskano zmniejszenie kosztu realizacji C o 19% (Arria V GZ, struktura 256_4x64_M13) oraz zmniejszenie maksymalnego czasu propagacji D: o 29% (Cyclone III, struktura 256_4x64_M11), o 26% (Arria II GX, struktura 256_4x64_M13), o 44% (Arria V GZ, struktura 256_4x64_M12), o 54% (Stratix III, struktura 256_2x128_2x64_M13).

5. Wnioski

Wyniki badań eksperymentalnych wykazały, że można zbudować 128-bitowe i 256-bitowe komparatory hierarchiczne, które dla pewnych rodzin układów FPGA mają lepsze parametry (koszt realizacji, maksymalny czas propagacji) od parametrów komparatorów z bezpośrednim wykorzystaniem funkcji *lpm_compare*. Przedstawiony sposób syntezy pozwala zmniejszyć koszt realizacji C oraz maksymalny czas propagacji D. Dla komparatorów 128-bitowych uzyskano zmniejszenie C o 13% oraz zmniejszenie D do 38% (w zależności od rodziny układów FPGA). W przypadku komparatorów 256-bitowych uzyskano zmniejszenie C o 19% oraz zmniejszenie D do 54% (w zależności od rodziny układów FPGA).

Do najlepszych struktur należą: 128_2x64_M12, 128_2x64_M13, 256_4x64_M11, 256_4x64_M12, 256_4x64_M13. Najbardziej efektywne są więc dwupoziomowe struktury hierarchiczne, w których do syntezy pierwszego stopnia wykorzystano równoległo-szeregowe metody M12, M13 (komparatory 128-bitowe) oraz M11, M12, M13 (komparatory 256-bitowe).

6. Literatura

[1] Parhami B.: Efficient hamming weight comparators for binary vec-tors based on accumulative and up/down parallel counters, IEEE Trans. Circuits Syst., vol. 56, no. 2, p. 167–171, 2009.

[2] Jarmolik W., Gruszewski M.: Nowy sposób projektowania uniwersalnego modułu do samotestowania układów hybrydowych, Elektronika, nr 4, s. 26-28, 2001.

[3] Cheng S.W.: Arbitrary Long Digit Sorter HW/SW Co-Design, Proceedings of IEEE Asia and South Pacific Design Automation Conference, p. 538-543, 2003.

[4] Suzuki H., Kim C. H., Roy K.: Fast tag comparator using diode partitioned domino for 64-bit microprocessor, IEEE Trans. Circuits Syst. I, vol. 54, no. 2, p. 322–328, 2007.

[5] Solov'ev V. V.: Proektirovanie cifrovyh sistem na osnove programmiruemyh logičeskih integral'nyh shem, Moskva, Gorâčaâ liniâ - Telekom, s. 636, 2001.

[6] Chuang P., Li D., Sachdev M.: A Low-Power High-Performance Single-Cycle Tree-Based 64-Bit Binary Comparator IEEE Transactions on Circuits And Systems-II: Express Briefs, vol. 59, no. 2, 2012.

[7] Deb S., Chaudhury S.: High-Speed Comparator Architectures for Fast Binary Comparison, Third International Conference on Emerging Applications of Information Technology (EAIT), p. 454-457, 2012.

[8] Deb S.: A Novel Architecture for Binary Comparison Using Time Division De-multiplexing Technique, Third International Conference on Emerging Applications of Information Technology (EAIT), p. 478-482, 2012.

[9] Hauser A., Chichester I.: High-Speed 64-Bit Binary Comparator using Two Stages, European Journal of Engineering and Innovation. vol. 11, 2013.

[10] Gruszewski M.: Metody syntezy komparatorów z wykorzystaniem języka Verilog w środowisku Quartus II, Elektronika, nr 1, s. 72-77, 2014.

otrzymano / received: 22.04.2014
przyjęto do druku / accepted: 02.06.2014 *artykuł recenzowany / revised paper*