Ernest JAMRO, Maciej WIELGOSZ, Kazimierz WIATR
KATEDRA ELEKTRONIKI, AKADEMIA GÓRNICZO-HUTNICZA
ACK CYFRONET, AKADEMIA GÓRNICZO-HUTNICZA

Realizacja operacji mnożenia o skróconej szerokości w układach FPGA

Dr inż. Ernest JAMRO

Ukończył studia na AGH na kierunku Elektronika oraz na University of Huddersfield (UK) na kierunku Elektronika i Telekomunikacja. Obronił pracę doktorską w 2001 roku na AGH na wydziale Elektrotechniki, Automatyki, Informatyki i Elektroniki. Aktualnie jest adiunktem w Katedrze Elektroniki na AGH. Jego zainteresowania naukowe to sprzętowa akceleracja obliczeń, niskopoziomowe przetwarzanie obrazów, sieci neuronowe.

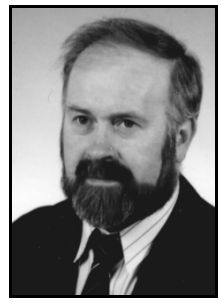
e-mail: jamro@agh.edu.pl

Mgr inż. Maciej WIELGOSZ

Ukończył studia na AGH (2005), wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki na kierunku Elektronika i Telekomunikacja. Obecnie jest doktorantem w Katedrze Elektroniki AGH i bierze czynny udział w pracach badawczych realizowanych w zespole rekonfigurowalnych systemów obliczeniowych. Jego zainteresowania naukowe dotyczą sprzętowej akceleracji obliczeń, kompresji obrazu i sieci neuronowych.

e-mail: wielgosz@agh.edu.pl

Prof. dr hab. inż. Kazimierz WIATR



Studia AGH Kraków (1980), dr nauk technicznych (1987), dr habilitowany (1999) i profesor (2002). Profesor zwyczajny na AGH w Krakowie oraz Dyrektor Akademickiego Centrum Komputerowego Cyfronet AGH. Prowadzone prace badawcze dotyczą systemów wizyjnych, systemów wieloprocesorowych, rekonfigurowanych systemów obliczeniowych i sprzętowych metod akceleracji obliczeń. Jest autorem trzech monografii, w tym najnowsza Akceleracja obliczeń w systemach wizyjnych wydana przez WNT w roku 2003.

e-mail: wiatr@agh.edu.pl

Streszczenie

Pełne mnożenie dwóch argumentów *n*-bitowych daje rezultat o szerokości *2·n*-bitów. W większości przypadków stosuje się mnożenie o skróconej szerokości gdzie np. dodatkowe *n* najmłodszych bitów wyniku jest odrzucane. Niniejszy artykuł prezentuje nową metodę kompensacji błędu obliczeń dla mnożenia o skróconej szerokości szczególnie wydajną w przypadku użycia układów FPGA. Podstawą proponowanej architektury jest podawanie na niewykorzystywane do tej pory wejście przeniesienia wybranych bitów argumentów wejściowych układu mnożącego.

Słowa kluczowe: układ mnożący, układy FPGA.

FPGA implementation of reduce-width multiplier

Abstract

The paper presents a novel metod of the error compensation for a reduce-width multiplier implemented in FPGAs. For a standard multiplier and the bit-width equal to *n* for both inputs, the output width is equal to *2·n*. In order to obtain a fixed-width multiplier, the *n*-LSBs of the output should be truncated. Lan-Da Van et. al. [1, 2] presented the error compensation method appropriate for ASIC, however, this method cannot be directly employed in FPGAs due to relatively high hardware resources and a different multiplier structure (compare Fig. 1 and Fig. 2). The main idea of the proposed error compensation method is to feed carry input directly with the selected bits of the multiplier input (see Fig. 4). The implementation results shown in Fig. 5 confirm the significant reduction of the truncation error, especially for the mean error which is close to zero. It should be noted that the error compensation circuit employs the normally unused carry-in input, therefore no additional FPGA resources are required by the proposed method.

Keywords: FPGA, fixed-width multiplier.

1. Wstęp

Operacja mnożenia jest jedną z podstawowych operacji arytmetycznych i dlatego liczba zasobów sprzętowych zajmowanych przez ten układ ma bardzo duże znaczenie. Operacja mnożenia jest zazwyczaj wykonywana z pełną szerokością bitową, to jest taką, dla której liczba bitów, na których zapisany jest wynik mnożenia jest równa sumie liczby bitów, na których zapisane są dwa argumenty wejściowe. Warto podkreślić, że w większości przypadków szerokość danych wejściowych i wyjściowych są takie same lub różnią się nieznacznie. Przykładem może być np. filtracja sygnału lub też operacja mnożenia zmiennoprzecinkowego. Jak zostanie pokazane w tym artykule redukcja szerokości danej wyjściowej już wewnątrz układu mnożącego może prowadzić do znaczącej redukcji zajmowanych zasobów układu FPGA kosztem zwiększonego błędu obliczeń. Błąd ten oraz liczba zajmowanych zasobów układu FPGA są uzależnione od liczby bitów, o które zostanie pomniejszony układ mnożący w stosunku do pełnego mnożenia.

Główną ideą tego artykułu jest autorski sposób korekcji błędu obcięcia – błędu obliczeń spowodowanego redukcją szerokości układu mnożącego już na etapie samego mnożenia. Zastosowanie tej korekcji w znaczący sposób zmniejsza błąd obliczeń nie wpływając przy tym na liczbę zajmowanych zasobów układu FPGA. Podobne metody korekcji błędów zostały zaproponowane w [1, 2], dotyczą one jednak układów mnożących zaimplementowanych w technologii ASIC i dlatego nie mogą być one bezpośrednio stosowane w układach FPGA.

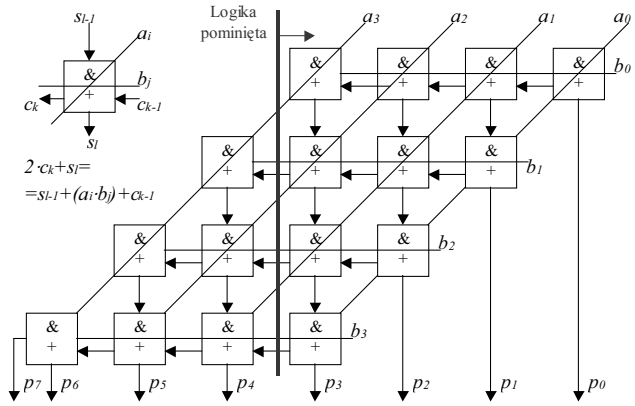
W niniejszym artykule zostanie omówiony układ mnożący implementowany w logice ogólnego przeznaczenia nazywanej również CLB (ang. Configurable Logic Blocks) [3]. Wprowadzenie w układach FPGA dedykowanego układu mnożącego [3] znacząco zmniejszyło zainteresowanie implementacją układu mnożącego przy użyciu logiki ogólnego przeznaczenia. Niemniej publikacja [4] dowodzi, że wbudowane układy mnożące wprowadzają znaczące opóźnienia w układzie i przez to w niektórych przypadkach nie są zalecane. Argumentem za użyciem logiki ogólnego przeznaczenia może być również fakt, że liczba dedykowanych układów mnożących jest ograniczona. Dlatego w niektórych projektach część układów mnożących (lub wybrane części tego samego układu mnożącego o dużej szerokości bitowej) może być implementowana przy użyciu zarówno dedykowanych układów mnożących jak i logiki ogólnego przeznaczenia, zwiększając tym samym funkcjonalność konkretnego układu FPGA.

2. Implementacja w technologii ASIC

Jedną z podstawowych metod implementacji układu mnożącego w technologii ASIC jest układ mnożący równoległy (ang. parallel-array multiplier) [5], którego schemat blokowy został przedstawiony na rys. 1. Warto podkreślić, że nie jest to optymalna metoda implementacji układu mnożącego w technologii ASIC [5], jednak ze względu na swoją prostotę to właśnie ona będzie tutaj rozpatrywana podobnie jak w [1, 2].

Istnieje wiele metod użycia takiego układu mnożącego dla stałej szerokości bitowej (szerokość bitowa argumentów wejściowych *n* jest taka sama jak szerokość bitowa wyjścia układu mnożącego). Standardowa metoda polega na implementacji całego układu

mnożącego przedstawionego na rys. 1 dla którego szerokość bitowa wyjścia wynosi $2 \cdot n$ (jest dwa razy większa niż szerokość bitowa wejść n). Następnie ogranicza się szerokość bitową wyjścia odrzucając n najmłodszych bitów wyniku; operacji tej zwykle towarzyszy operacja zaokrąglania. Metoda ta będzie dalej nazywana pełnym mnożeniem i jest ona najdokładniejsza jeśli chodzi o błąd obliczeń i zarazem najgorsza jeśli chodzi o ilość zajmowanych zasobów.



Rys. 1. Układ mnożący równoległy dla n=4
Fig. 1. Parrallel array multiplier for the input width n=4

Inną metodą implementacji operacji mnożenia ze stałą szerokością jest struktura z bezpośrednim odcięciem (ang. direct truncated structure [1, 2]) przedstawiona na rys. 1, dla której cała logika na prawo od bitu p_n (najmłodszego bitu wyjścia, który jest wyprowadzany na zewnątrz) nie jest implementowana. Struktura ta charakteryzuje się najmniejszą zajmowaną powierzchnią układu scalonego z jednej strony, ale największym błędem obliczeń z drugiej strony. Analizując rys. 1 można zauważyć, że struktura ta zajmuje niewiele mniej niż połowę zasobów pełnego układu mnożącego. Potwierdza to publikacja [1], według której struktura z bezpośrednim odcięciem zajmuje od 44% (dla n=6) do 49% (dla n=16) powierzchni pełnego układu mnożącego w technologii ASIC.

Pomiędzy strukturą pełnego mnożenia a strukturą bezpośredniego odcięcia istnieją struktury pośrednie, dla których w porównaniu ze strukturą bezpośredniego odcięcia, implementowane jest w dodatkowych bitów na prawo od bitu p_n (zob. rys. 1). W konsekwencji bezpośrednio w wyniku mnożenia otrzymuje się $(n+w)$-bitów wyniku, który następnie jest zaokrąglany do n bitów poprzez odrzucenie w najmłodszych bitów. Warto pokreślić że struktury pośrednie są tożsame z układem pełnego mnożenia dla $w=n$, oraz z układem bezpośredniego odcięcia dla w=0. W konsekwencji zwiększając parametr w możliwe jest uzyskanie większej dokładności obliczeń kosztem większych zajmowanych zasobów.

Publikacje [1, 2] prezentują metodę zmniejszenia błędu obliczeń stosując dodatkową logikę kompensacji błędu. Logika ta składa się w pierwszym przybliżeniu z bramek AND i OR dołączonych do wejść przeniesienia wejściowego w miejscach odcięcia. Korekcja błędu σ zgodnie z równaniem (34) [2] jest wyrażona wzorem:

$$\sigma = a_{n-2} \cdot b_1 + a_{n-3} \cdot b_2 + a_{n-4} \cdot b_3 + \ldots + a_1 \cdot b_{n-2} \ (+1) \qquad (1)$$

Warto podkreślić, że kolejne iloczyny logiczne $a_i \cdot b_j$ użyte w równaniu (1) są to iloczyny bitów, które byłyby użyte w układzie mnożącym dla szerokości w większej o 1, czyli dla punktu odcięcia przesuniętego o jeden bit w prawo (zob. rys. 1). Jednak byłyby one brane z wagą ½ (dotyczą one kolejnego młodszego bitu). Tutaj natomiast są one brane z wagą 1, co kompensuje fakt, że wraz z redukcja szerokości pomijane są również kolejne iloczyny bitowe o wadze ¼, itd.

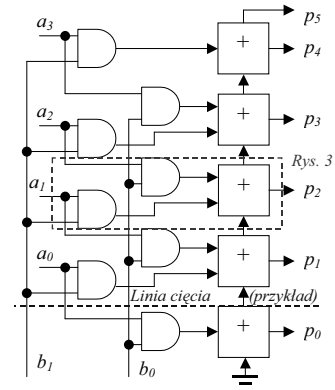
W strukturze ASIC dodatkowe bramki AND i OR potrzebne do zaimplementowania równania (1) zajmują dużo mniej powierzchni układu scalonego niż pełne sumatory, dlatego zastosowanie tej metody przynosi wymierne korzyści. Niestety w przypadku układu FPGA umiejscowienie dodatkowych bramek AND i OR wymagałoby użycia dodatkowej pamięci LUT, co jest mniej więcej równoważne z punku widzenia zajmowanych zasobów sprzętowych, z przesunięciem punktu odcięcia o jeden bit w prawo (zwiększeniem w o 1). Ponadto struktura układu mnożącego implementowanego w układach FPGA (co zostanie pokazane w następnym rozdziale) wygląda inaczej niż na rys. 1. Podsumowując technika prezentowana w [1, 2] nie jest optymalnym rozwiązaniem w przypadku implementacji w układach FPGA.

3. Implementacja w układach FPGA

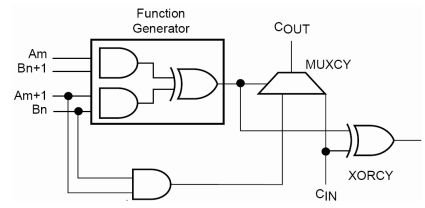
Struktura układu mnożącego implementowanego w układach FPGA została zaprezentowana np. [6] i powtórzona na rys. 2 i rys. 3 dla układu mnożącego 2×4-bity. W układzie tym w ramach jednego elementu logicznego LE (Logic Element - składającego się w przybliżeniu z pamięci LUT, logiki przeniesienia arytmetycznego oraz przerzutnika) implementowana jest logika (rys. 3):

$$2 \cdot C_{OUT} + P_i = A_m \cdot B_{n+1} + A_{m+1} \cdot B_n + C_{IN} \qquad (2)$$

gdzie: A_m, A_{m+1} – kolejne bity argumentu wejściowego A, B_{n+1}, B_n – kolejne bity argumentu wejściowego B, C_{IN}, C_{OUT} – wejście i wyjście przeniesienia, P_i- i-ty bit wyniku mnożenia.



Rys. 2. Struktura układu mnożącego 2×4-bity [6]
Fig. 2. Block diagram of the 2×4-bit multiplier [6]



Rys. 3. Zaznaczony prostokąt z Rys. 2 i jego implementacja w strukturze Virtex.
Fig. 3. The selected rectangular form Fig. 2 and its implementation in Virtex

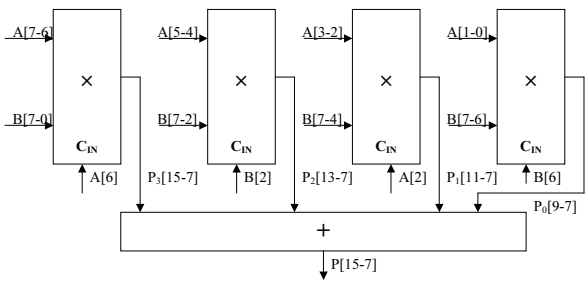
Aby uzyskać układ mnożący o szerokości $n \times n$ bitów należy użyć $\lceil n/2 \rceil$ przedstawionych powyżej $2 \times n$-bitowych układów mnożących oraz odpowiednio zsumować wyniki pośrednie P.

4. Zaproponowana kompensacja błędu odcięcia

Podobnie jak w przypadku układu mnożącego przedstawionego na rys. 1, możliwa jest redukcja szerokości bitowej układu mnożącego implementowanego w układach FPGA, poprzez brak implementacji logiki dla bitów młodszych od P_{n-w}. W przypadku rys. 2, będzie to polegało na braku implementacji elementów logicznych znajdujących się poniżej (młodszych) bitu P_{n-w} (na rys. 2 jest to zaznaczone kreską poniżej bitu P_1, która wyznacza granicę implementacji).

Główna idea niniejszego artykułu polega na lekkiej modyfikacji zredukowanego układu mnożącego poprzez wykorzystanie przeniesienia wejściowego C_{IN}, które na rys. 2 jest podłączone do masy. Analizując równanie (1) można zauważyć, że przy równym rozkła-

dzie prawdopodobieństwa wystąpienia stanu 0 i 1 dla zmiennych a_i oraz b_j, średnia wartość wyrażenia $a_{n-2}\cdot b_1 + a_{n-3}\cdot b_2$ jest równa ½. Z punktu widzenia prawdopodobieństwa można więc zastąpić wyrażenie $a_{n-2}\cdot b_1 + a_{n-3}\cdot b_2$ jednym z wejść: a_{n-2}, b_1, a_{n-3} lub b_2, czyli możliwe jest zastąpienie całego wyrażenia $a_{n-2}\cdot b_1 + a_{n-3}\cdot b_2$ pojedynczym dowolnym bitem wchodzącym w skład tego wyrażenia. Z punktu widzenia sprzętowej implementacji powyższe zastąpienie przynosi zdecydowane zmniejszenie zajmowanych zasobów, ponieważ wystarczy doprowadzić do nieużywanego do tej pory wejścia przeniesienia C_{IN} odpowiedni bit danej wejściowej. W konsekwencji proponowana kompensacja błędu redukcji szerokości układu mnożącego nie wymaga żadnych dodatkowych zasobów sprzętowych. Warto podkreślić, że odbywa się to kosztem zwiększonego błędu obliczeń w porównaniu z użyciem pełnego wyrażenia $a_{n-2}\cdot b_1 + a_{n-3}\cdot b_2$.



Rys. 4. Schemat blokowy proponowanego układu mnożącego dla w=1
Fig. 4. Block diagram of the proposed multiplier for w=1, four blocks of the multiplier are similar as in Fig. 2

Przykładowy schemat zaproponowanego układ mnożącego został przedstawiony na rys. 4 dla n=8 oraz w=1. W ramach tego układu użyto czterech bloków mnożących $2\times k$-bitowych, których budowa jest podobna jak na rys. 2. W ramach proponowanej korekcji błędu odcięcia, do wejścia przeniesienia C_{IN} każdego bloku mnożącego $2\times k$ podawany jest tylko jeden wybrany bit danej wejściowej A lub B, Jeden z najstarszych bitów, który byłby wykorzystywany dla parametru w większego o 1. Warto zwrócić uwagę jak na rys. 4 podłączone są wejścia przeniesienia, które korygują błąd odcięcia. Wejścia te to kolejno bity *A[6], B[2], A[2]* oraz *B[6]*.

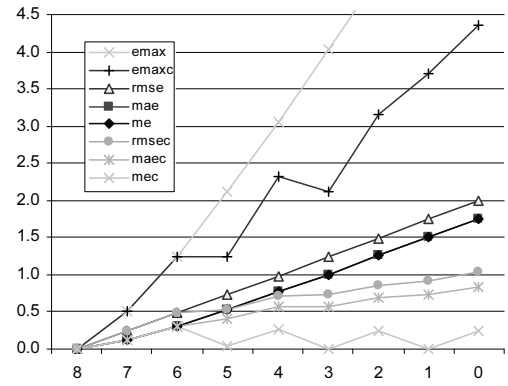
5. Wyniki implementacji

W ramach pomiaru błędu obliczeń wykorzystywane są cztery definicje błędów:

- ME (Mean Error) – błąd średni: $ME = \frac{1}{N}\sum_{i=1}^{N} e_i$
- MAE (Mean Absolute Error) – błąd średni z wartości bezwzględnej błędu: $MAE = \frac{1}{N}\sum_{i=1}^{N} |e_i|$
- RMSE (Root Mean Square Error) – błąd średniokwadratowy: $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}$
- Emax – błąd maksymalny: $Emax=MAX(|e_i|)$,

gdzie e_i jest różnicą pomiędzy wartością poprawną (wynik dla pełnego układu mnożącego: w=n) a wartością otrzymaną dla i-tej próbki. Podczas obliczeń były brane wszystkie możliwe kombinacje wejść i wyjść, co dla n=8, daje N= $2^{2\cdot n}$= 2^{16} możliwych kombinacji.

Rys. 5 przedstawia wartości wcześniej zdefiniowanych błędów dla n=8 i dwóch układów: bez i z kompensacją błędu odcięcia. W celu rozróżnienia wyników, dla układu z kompensacją błędu dodano końcówkę c (np. emax**c**). Dla przejrzystości wykresu etykiety są umieszczone w kolejności od największego błędu dla w=0. Powyższy wykres został opracowany dla mnożenia bez znaku dlatego nie może być w prosty sposób porównany z publikacjami [1, 2]. Waga błędu e_i jest podana w ULP (ang. Unit Last Place) czyli ma wagę najmłodszego wyprowadzonego bitu P_{n-w}, co jest równoważne 2^{n-w} ULP dla pełnego sumatora (w=n). Na przykład, waga błędu dla w=4 jest 16 razy mniejsza niż dla w=0. Warto zwrócić uwagę, że dla braku kompensacji, błąd średni *ME* i błąd średni modułu *MAE* są takie same, czyli wartość poprawna jest zawsze niemniejsza od otrzymanej.



Rys. 5. Wykres błędu [ULP] bez i z korekcją dla różnej wartości parametru w
Fig. 5. Errors [ULP] for different values of the parameter w

Analizując wartości błędu bez i z zaproponowaną korekcją błędu można zauważyć zdecydowaną redukcję błędu. Na przykład dla w=0, błąd maksymalny bez kompensacji błędu wynosi *Emax*= 7.0 (wartość poza wykresem) natomiast z kompensacją *Emaxc*= 4.35. Redukcja błędu jest szczególnie widoczna dla błędu średniego *ME*, który dla w=0, przy braku kompensacji błędu wynosi 1.75 oraz 0.25 z korekcją. Dla różnych w przy zastosowaniu korekcji błędu, błąd średni *MAC* jest w granicach 0-0.25. Warto podkreślić, że błąd średni *ME* w niektórych zastosowaniach ma kluczowe znaczenie, ponieważ w przypadku kiedy błąd średni jest równy zero suma poszczególnych błędów się równoważy.

6. Wnioski

W niemniejszym artykule przedstawiono nową metodę kompensacji błędu redukcji szerokości układu mnożącego dedykowaną dla układów FPGA. Zastosowanie autorskiej kompensacji błędu zmniejsza zdecydowanie błąd obliczeń bez zajmowania dodatkowych zasobów sprzętowych. Na szczególną uwagę zasługuje redukcja błędu średniego który waha się w granicach 0-0.25 ULP (Unit Last Place).

Publikacja finansowana ze środków na naukę MNiSW w roku 2009.

7. Literatura

[1] Lan-Da Van, Chih-Chyau Yang: Generalized Low-Error Area-Efficient Fixed-Width Multipliers, IEEE Transactions on Circuits and Systems, VOL. 52, NO. 8, pp. 1608-1619, August 2005.
[2] Lan-Da Van, Shuenn-Shyang Wang, Wu-Shiung Feng: Design of the Lower Error Fixed-Width Multiplier and Its Application, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 47, no. 10, pp. 1112-1118, OCTOBER 2000.
[3] Xilinx, Virtex-4 Family Overview, www.xilinx.com, DS112 (v3.0) September 28, 2007.
[4] J. Poldre and R. Tammemae: Reconfigurable multiplier for Virtex FPGA family, Int. Workshop on Field- Programmable Logic and Applications, Glasgow, Scotland, UK, pp. 359-364, Aug. 30 –Sept. 1, 1999.
[5] A. R. Omondi, Computer Arithmetic Systems: Algorithms, Architecture and Implementation, Prentice-Hall International, 1994.
[6] S. Elzinga, J. Lin, V. Singhal: Design Tips for HDL Implementation of Arithmetic Functions, Xilinx Application Note XAPP215 (v1.0) June 28, 2000.

Artykuł recenzowany