Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

ARYTMETYKA RESZTOWA W CYFROWYM PRZETWARZANIU SYGNAŁÓW

Janusz Jabłoński

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: J.Jablonski@iie.uz.zgora.pl

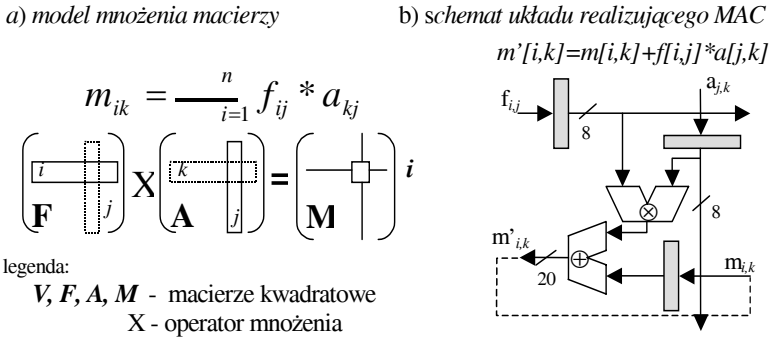
STRESZCZENIE

Mechanizmy zwiększenia wydajności przetwarzania informacji w urządzeniach cyfrowych są ciągle aktualnym tematem prac badawczych w wielu ośrodkach naukowych. Proponowane rozwiązania opierają się na akceleracji sprzętowej oraz formule „divide and conquere". W zgodzie z tą formułą jest wykorzystanie resztowej reprezentacji liczb (*ang. Residue Number System, RNS*) w sprzętowej realizacji obliczeń. Efektem wykorzystania *RNS* w *DSP* (*ang. Digital Signal Processing*) jest ograniczenie wpływu propagacji przeniesień w operacjach arytmetycznych, zwiększenie stopnia zrównoleglenia operacji i może być zwiększenie wydajności przetwarzania. Jednak, zapewnienie jednoznacznej reprezentacji wyników oraz pokrycia zakresu wymaga rozwiązania szeregu problemów związanych z konstrukcją modułów tworzących bazę *RNS*. Prezentowany materiał stanowi przegląd problemów związanych z konstruowaniem układów arytmetyki modularnej o cechach umożliwiających automatyzację zmian zakresu przy zachowaniu możliwości uzyskania wzrostu wydajności przetwarzania.

1. WPROWADZENIE

Przetwarzanie informacji oparte jest na kodowaniu, które umożliwia precyzyjne przedstawienie danych w postaci cyfr określonego alfabetu. Jeżeli cyframi są liczby to przetwarzanie informacji polega na wyznaczeniu odpowiedzi, będącej wynikiem przeprowadzonych obliczeń. Przez analogię do modelowania matematycznego a w oparciu o gromadzone w postaci liczb informacje oraz odpowiedni i właściwy dla danego zjawiska aparat matematyczny możliwe jest symulacyjne przedstawienie przebiegu większości przemian fizycznych, chemicznych oraz innych zjawisk zachodzących w otaczającym świecie [3]. Obecnie przetwarzanie informacji realizowane jest przez systemy cyfrowe, przy czym szczególnie w realizacji algorytmów: kompresji danych, kryptografii, filtracji cyfrowej, przetwarzania obrazów wymagana jest duża wydajność obliczeniowa. W szczególności telekomunikacja jest przykładem dziedziny, w której rozwój zdeterminowany jest przez urządzenia cyfrowe wymagające dużych mocy obliczeniowych. W ogólnym przypadku większość zadań *DSP* można sklasyfikować jako splot, korelacja lub transformacja, realizowane jako forma mnożenia akumulacyjnego (*ang. Multiple and Accumulate, MAC*) [4]. Schemat mnożenia macierzy wykorzystujący *MAC* jako operacje dominujące przedstawia

rys. 1. Uwzględniając model Winograda układu cyfrowego czas wykonania operacji, również arytmetycznych, uzależniony jest od rozmiaru argumentów i zwiększenie wydajności przetwarzania można uzyskać przez: ograniczanie rozmiaru argumentów, zrównoleglenie wykonywanych działań oraz implementację przetwarzania potokowego[1][6]. Układowe dodawanie jak również mnożenie realizowane jest przez układy cyfrowe, więc parametrem mającym wpływ na czas wykonania operacji jest liczba *n* pozycji bitowych argumentów [9]. Przy czym dodawanie jest podatne na wykorzystanie mechanizmów zrównoleglenia i dla sumatorów prefiksowych przeliczeniowa liczba bramek podstawowych wynosi $2\log_2 n$ [7][9]. Natomiast układy mnożące są mniej podatne na zrównoleglenie a minimalna liczba elementów na ścieżce krytycznej wynosi $2(n+\log n)$, dla mnożenia matrycowego oraz $4\left\{\log_2 \frac{2n}{6}\right\}+3\log_2 n$, dla mnożenia równoległego [1].



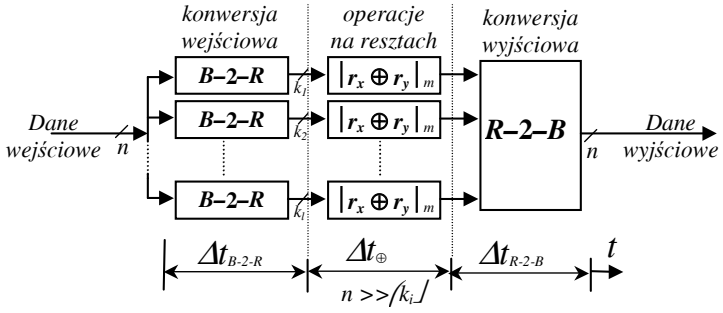
Rys. 1. Model algorytmu wykorzystujący schemat mnożenia akumulacyjnego MAC

Wydajność potokowych realizacji algorytmów *DSP*, rozumiana jako ilość wygenerowanych danych w czasie zależy znacząco od przepustowości i głębokości potoku [1]. Głębsza dekompozycji operacji dodawania i mnożenia w potokowego powoduje skrócenie czasu trwania etapu, jednak zwrotne sprzężenie danych (występujące w mnożeniu macierzy, rys. 1b powoduje znaczne opóźnienia wynikające z większego stopnia komplikacji układów sterujących potokiem. W efekcie nadmiernie rozbudowany układ sterowania stanowi barierę w uzyskaniu wzrostu wydajności [1][6].

2. ARYTMETYKA RESZTOWA W OBLICZENIACH

W systemie resztowym reprezentowane są wartości całkowite a reprezentacją liczby *X* jest wektor $R_X = \{r_{x_1}, r_{x_2}, ..., r_{x_i}\}$, gdzie $r_{x_i} = |X|_{m_i} = X \bmod m_i$. Jeżeli moduły $m_i \in N$ spełniają warunek $NWD(m_i, m_j) = 1$ to wektor $\mathbf{M} = \{m_1, m_2, ..., m_l\}$ nazywany jest bazą systemu

resztowego. Wartość $W = \prod_{i=1}^{l} m_i$ nazywana jest zakresem *RNS* w bazie **M**. Zakres W zapewnia, że dla każdego $X < W$ reprezentacja resztowa R_X jest jednoznaczna. Arytmetyka resztowa nazywana jest „arytmetyką bez przeniesień", ponieważ w *RNS* operacje $\oplus \in \{+,-,*\}$ realizowane są niezależnie na resztach, w poszczególnych kanałach resztowych [5][8]. Zatem, w *RNS* wynik Z operacji $X \oplus Y$ reprezentuje wektor R_Z reszt $\{r_{z_1}, r_{z_2}, ..., r_{z_l}\}$, gdzie $r_{z_i} = \left| r_{x_i} \oplus r_{y_i} \right|_{m_i}$. Wynika stąd, że dla zapewnienia jednoznaczności wyników ($Z < W$) oraz zmniejszenia rozmiaru bitowego argumentów operacji arytmetycznych $(r_{x_i} \oplus r_{y_i})$, którymi w *RNS* są wartości reszt, wystarczy $l = 2$ przy operacjach „+" oraz l =3 przy mnożeniu (czytaj odpowiednio: dwa i trzy moduły względnie pierwsze) [2]. Można zauważyć również, że wykorzystanie *RNS* i rozszerzenie bazy **B** o kolejne moduły m_i wpływa na zmniejszenie rozmiaru bitowego reszt, umożliwiając w ten sposób uzyskanie większej przepustowości potoku oraz większej wydajności przetwarzania. Model systemu cyfrowego w/g Głószkowa wskazuje, że jednostka realizująca obliczenia jest ważnym, lecz tylko jednym z podsystemów przetwarzania danych. Natomiast, nie wszystkie operacje wymagane w realizacji algorytmów obliczeniowych oraz sterowania są łatwe do realizacji w *RNS*. W szczególności do operacji trudnych (czasochłonnych) w *RNS* należą: detekcja znaku, dzielenie, porównanie wartości oraz zmiana zakresu [8]. Tak, więc, aby wykorzystać zalety arytmetyki resztowej w przetwarzaniu, należałoby korzystać z dwóch nie zależnych jednostek przetwarzania danych. Jednak, ze względów ekonomicznych rozwiązanie takie nie jest praktykowane. Innym, znajdującym szereg praktycznych zastosowań, rozwiązaniem jest konwersja wszystkich danych biorących udział w przetwarzaniu do systemu resztowego oraz odwrotna konwersja wyników. Uwzględniając powyższe rozważania schemat kolejności etapów dla potoku przetwarzania z wykorzystaniem *RNS* przedstawia rys. 2.



Rys. 2. Etapy potoku z wejściową i wyjściową konwersją danych

Ze względu na wydajności kodowania liczb oraz szybkość obliczeń uzasadnione jest wykorzystanie binarnej reprezentacji liczb [4]. Wówczas, konwersja do *RNS* polega na przejściu od reprezentacji binarnej do resztowej (ang. *Binary to Residue*, *B–2–R*) natomiast

konwersja wyników polega na przejściu z reprezentacji resztowej do binarnej (ang. *Residue* to *Binary, R–2–B*).

3. KONWERSJA I SKALOWANIE ZAKRESU SYSTEMU RESZTOWEGO

Konwersja wejściowa oraz wyjściowa wymaga zwiększenia głębokości potoku wprowadzając dodatkowe opóźnienia (rys. 2). Wynika stąd, że możliwe uzyskanie zwiększenia wydajności przetwarzania w *RNS* jest uwarunkowane czasem związanym z realizacją etapów konwersji, przy czym dla wydajności potoku korzystnie jest, jeżeli $\Delta t_{B-2-R} \cong \Delta t \oplus \cong \Delta t_{R-2-B}$. Ponadto, najkorzystniejsze rezultaty w przyspieszeniu obliczeń uzyskane zostaną, jeżeli zakres W systemu resztowego zostanie dostosowany do rozmiaru wyników obliczeń [2]. W *RNS* możliwe są trzy metody postępowania w celu uzyskania zmiany zakresu: zmiana liczby modułów m_i, zmiana ich wartości lub zmiana liczby i wartości modułów [2][4][5]. Optymistyczna skala redukcji zakresu działań odwrotnie proporcjonalna do liczby modułów jest nieosiągalna, również szybkość działań arytmetycznych nie maleje odwrotnie proporcjonalnie do zakresu argumentów [2]. Z tego względu wybór modułów wymaga szczegółowej analizy uwzględniającej, że zasadniczy wpływ na złożoność i w konsekwencji czasochłonność działań ma liczba i wartość modułów bazy. Synteza konwersji wyjściowej najczęściej oparta jest na Chińskim Twierdzeniu o Resztach (*CRT*) oraz pamięciach wykorzystanych jako tablice odniesień z wyznaczonymi wcześniej wartościami odpowiednich współczynników. Podobne, oparte na pamięciach rozwiązania można wykorzystać w realizacji konwersji wejściowej. Jednakże, rozwiązania takie wymagają czasochłonnego wyznaczania wartości współczynników oraz uaktualnienia zawartości pamięci każdorazowo przy zmianie zakresu *RNS* i restarcie lub awarii systemu. Ponadto, dla modułów o liczbie pozycji bitowych $k > 7$, zauważalne są niekorzystne efekty związane z wykładniczo rosnącym zapotrzebowaniem na pamięć. Zatem, wykorzystanie *RNS* oraz układów konwersji wejściowej i wyjściowej bazujących na tablicach odniesień nie zapewnia wzrostu szybkości przetwarzania. Szczególnie, jeżeli często zmieniany jest zakres reprezentacji liczb. Uwzględniając powyższe rozważania poszukiwane są metody realizacji konwersji oraz działań modulo oparte na blokach kombinacyjnych o regularnych strukturach. Do rozwiązań tych można zaliczyć układy arytmetyczne ze szczególnym uwzględnieniem szybkich sumatorów prefiksowych oraz matrycowych układów mnożących. Jednakże, chcąc wykorzystać zalety układów kombinacyjnych należy umiejętnie wybierać moduły m_i bazy **B** systemu resztowego. Wybór liczby modułów wpływa z jednej strony na możliwość zrównoleglenia działań, z drugiej strony na złożoność konwersji. Przy nadmiernym zwiększaniu liczby modułów można zaobserwować intensyfikację dwóch niekorzystnych zjawisk. Pierwsze, wynika z niemożliwości wyboru dużej liczby względnie pierwszych modułów o zbliżonej wartości, przez co może się okazać, że po dołączeniu kolejnego modułu nie nastąpi znacząca redukcja rozmiaru najszerszego kanału resztowego. Drugi efekt,

związany jest z procedurami konwersji odwrotnej. Analiza rozwiązania wynikającego z *CRT* prowadzi do spostrzeżenia, że poszczególne „jedynki resztowe" są iloczynami tylu liczb ile jest modułów m_i, które w reprezentacji pozycyjnej są postaci $\beta^{k_i} \pm \alpha$, więc każda z nich ma wartość wyznaczoną przez sumę wielokrotności iloczynów potęg β^{k_i} [2]. W efekcie, złożoność konwersji odwrotnej rośnie z kwadratem liczby modułów. W praktycznych zastosowaniach, z uwagi na problemy z dobraniem dużej liczby względnie pierwszych modułów o zbliżonych wartościach, wystarczy przeanalizować systemy *RNS* wykorzystujące 2 do 5 modułów o konstrukcji podatnej na zmiany zakresu oraz skalowanie układów biorących udział w przetwarzaniu.

4. SYNTEZA UKŁADOWA DLA ARYTMETYKI MODULARNEJ

Podstawowe działania arytmetyczne to dodawanie oraz mnożenie a najszybsze ich układowe realizacje oparte są odpowiednio; na sumatorach prefiksowych[7][9] i potokowej realizacji matrycy mnożącej lub równoległym układzie mnożącym [1][6]. W pracy [2] zaproponowano konstruowanie *RNS* w oparciu o moduły $m_i = 2^k \pm \alpha$. Opracowano nowe układy generowania reszt modulo $(2^k \pm 1)$ oraz zaproponowano wykorzystanie szybkich sumatorów prefiksowych z korekcją w układach generowania reszt, dodawania oraz konwersji odwrotnej. Wykazano możliwość konstruowania *RNS* z modułami typu $(2^k \pm 3, 2^k \pm 1, 2^k)$ oraz zbadano ich właściwości pod kątem podatności na skalowanie i możliwości implementacji takich systemów w strukturach *FPGA*. Ponadto, dla proponowanych modułów opracowano i przedstawiono autorskie metody syntezy konwersji wejściowej (*B*–2–*R*) oraz konwersji wyjściowej (*R*–2–*B*). Dla *RNS* w bazie $\{2^k-1, 2^k, 2^k+1\}$, opracowano nową metodę i schematy syntezy konwersji wyjściowej, zaproponowane rozwiązania układowe umożliwiają dynamiczne skalowanie zakresu na wszystkich etapach przetwarzania (generowanie reszt, dodawanie i mnożenie oraz konwersja odwrotna). Proponowane układy to rozwiązania o najmniejszej liczbie poziomów logicznych na ścieżce krytycznej [10].

5. ZAKOŃCZENIE

Teoretycznie, wykorzystanie *RNS* o dużej liczbie modułów powoduje zmniejszenie rozmiaru argumentów obliczeń i powinno powodować proporcjonalne zwiększenie wydajności przetwarzania. Jednak w praktyce, ze względu na zmiany oraz konwersji pomiędzy systemami reprezentacji liczb, osiągany wzrost wydajności jest mniejszy [2][5][8]. Dla analizowanych i implementowanych w *FPGA* rozwiązań, uzyskano wyniki wskazujące

możliwość zwiększenia przepustowości potoku przetwarzania dla typowych w *DSP* obliczeń od 10 do 30% [2]. Dzięki wykorzystaniu najszybszych rozwiązań układowych można przypuszczać, że podobne lub korzystniejsze rezultaty są możliwe do uzyskania również z wykorzystaniem innych niż *FPGA* technologii. Jednakże, w kontekście podatności układów arytmetyki resztowej na zmiany zakresu, modelowanie autorskich rozwiązań dostosowano do wykorzystania syntezowalnego podzbioru języka opisu sprzętu *VHDL*, powoduje to znaczne ułatwienie w implementacji skalowalnych układów arytmetyki modularnej. Szczególnie interesująca jest możliwość rekonfigurowania *FPGA* bez wykorzystania zewnętrznych narzędzi *CAD*, możliwa do realizacji w zintegrowanych systemach cyfrowych typu *FPSLIC* (ang. *Field Programmable System Level Integrate Circuits*). Spodziewane kierunki dalszych badań to wypracowanie metod oraz zaprojektowanie systemu dynamicznej rekonfiguracji zasobów logiki programowalnej. Rekonfiguracja ukierunkowana będzie na wykorzystanie arytmetyki resztowej oraz dostosowanie logiki programowalnej do bieżącego zakresu obliczeń i wymaganego czasu realizacji algorytmów obliczeniowych. W tym celu opracowane zastaną metody autokonfiguracji zasobów *FPGA* oraz automatycznej syntezy konwersji dla *RNS* z modułami typu $(2^k \pm 1)$, $(2^k \pm 3)$ oraz uogólnienie jej dla $(2^k \pm \alpha)$. Ponadto, wypracowane metody autokonfiguracji zasobów *FPGA* w zintegrowanych systemach cyfrowych, mogą stanowić podstawy do prac zmierzających w kierunku projektowania odpornych na błędy oraz uszkodzenia systemów sprzętowo programowych.

LITERATURA

[1] J. Biernat: *Metody i układy arytmetyki komputerowej,* Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław 2001

[2] J. Jabłoński: *Seryjne wykonywanie operacji arytmetycznych w strukturach reprogramowalnych*, rozprawa doktorska, Politechnika Wrocławska, Wrocław 2003

[3] Z. Kierzkowski: *Maszyny matematyczne i ich zastosowania*, Wydawnictwo uczelniane Politechniki Poznańskiej, Poznań 1966.

[4] B. Pochopień: *Arytmetyka systemów cyfrowych*, Wydawnictwo Politechniki Śląskiej, Gliwice 2000

[5] N.S. Szabo, R.I. Tanaka: *Residue arithmetic and its applications to computer technology*, McGraw-Hill, New York, 1967

[6] W. Stallings: *Organizacja i architektura systemu komputerowego*, Wydawnictwo Naukowo Techniczne, WNT Warszawa 2000

[7] V. Tchoumatchenko: *Modélisation, Architecture et Outils de Synthése pour Additionneurs Rapides*, Ph.D thesis, de I'Institut National Polytechnique de Grenoble, Grenoble, 1998

[8] Z. Ulman: *Resztowe systemy liczbowe z quasi podstawą*, Zeszyty Naukowe Politechniki Gdańskiej, Elektryka Nr 83, Gdańsk, 1998

[9] R. Zimmerman: *Binary Adder Architectures for Cell—Based VLSI and their synthesis*, Ph.D thesis, Swiss Federal Institute of Technology 1997