Wojciech SUŁEK
POLITECHNIKA ŚLĄSKA

Konfigurowalny dekoder kodów LDPC implementowany w układzie FPGA

Dr inż. Wojciech SUŁEK

Autor jest pracownikiem naukowym Instytutu Elektroniki Politechniki Śląskiej, a jego zainteresowania naukowe są związane głównie z dziedziną telekomunikacji. Prowadzone przez autora prace badawcze dotyczą zwłaszcza zagadnień efektywnej implementacji sprzętowej algorytmów przetwarzania sygnałów oraz systemów telekomunikacyjnych, w szczególności modułów nowoczesnego kodowania kanałowego.

e-mail: wsulek@polsl.pl

Streszczenie

Kody LDPC są jednymi z najlepszych znanych klas kodów nadmiarowych, służących do korekcji błędów w kanale telekomunikacyjnym. W niniejszej pracy zaprezentowano opisany w języku VHDL konfigurowalny dekoder podklasy kodów LDPC zorientowanych na efektywną sprzętową implementację. Możliwe jest dostosowanie dekodera dla dowolnego kodu LDPC ze zdefiniowanej podklasy, jak również konfiguracja pewnych parametrów dekodera decydujących o jego własnościach strukturalnych oraz własnościach korekcyjnych systemu. W artykule przedstawiono możliwości konfiguracji dekodera oraz wyniki implementacji: zasoby strukturalne oraz przepustowość dla kilku wybranych kodów.

Słowa kluczowe: kody blokowe, kodowanie kanałowe, kody LDPC, dekodowanie iteracyjne, dekoder LDPC.

Configurable LDPC decoder implemented in FPGA device

Abstract

The group of Low-Density Parity-Check (LDPC) codes is one of the best known error correcting coding methods that are capable of achieving very low bit error rates at code rates approaching Shannon's channel capacity limit. The article concerns the configurable decoder for a subclass of LDPC codes that are implementation oriented. The decoder has a form of synthesizable VHDL description. It can be adjusted for decoding any code from defined subclass, called Architecture Aware LDPC (AA-LDPC). Configuration of some decoder parameters (message calculating algorithm, message wordlength) is possible as well. These parameters affect decoder structural properties and on the other hand – error correcting performance of the coding system. A number of modifications in the VHDL source code are required to adjust the decoder to the particular AA-LDPC code. These modifications can be made automatically by a software that has been created using Matlab tool. The user needs only to specify the parity check matrix that has architecture-aware structure as well as to specify other parameters of the decoder, such as: message wordlength, maximum number of iteration, the number of computing units (SISO) and the SISO message update (sub-optimal) algorithm. Based on these parameters, automatic generation of synthesizable VHDL description can be performed by the software tool that has been created. The decoder is implemented with the Xilinx VirtexII FPGA device. The simulation environment, making use of the hardware decoder is a base of the platform for fast simulation of the developed LDPC coding systems performance. In this paper we present mainly the decoder reconfiguration methods. Implementation results: structural resources and decoder throughput for a couple of different codes are presented as well.

Keywords: block codes, channel coding, LDPC codes, iterative decoding, LDPC decoder.

1. Wstęp

Zastosowanie kodowania LDPC pozwala osiągać bardzo niskie prawdopodobieństwo błędu transmisji przez zakłócony kanał telekomunikacyjny, przy stosunku mocy sygnału do mocy szumu (SNR) bliskim teoretycznej granicy Shannona. Kody LDPC są pod tym względem najlepszymi ze znanych klas kodów blokowych. Idea kodowania LDPC została przedstawiona po raz pierwszy w latach 60-tych ubiegłego wieku przez R.G. Gallagera [1], jednakże wysokie wymagania na moc obliczeniową modułów kodowania i dekodowania powodowały, że przez lata praktyczne wykorzystanie kodów LDPC było bardzo ograniczone. Wzrost mocy obliczeniowej współczesnych układów cyfrowych spowodował ponowne zainteresowanie badaczy oraz organizacji standaryzacyjnych rozwojem metod kodowania LDPC [2]. Są one między innymi wdrażane do zastosowania w standardach transmisji sygnału telewizji cyfrowej oraz bezprzewodowych sieciach komputerowych nowych generacji.

W przypadku kodów blokowych (których klasą są kody LDPC), ciąg bitów przesyłany w systemie transmisji dzielony jest na równe bloki K bitów informacyjnych $\mathbf{u}=[u_1,u_2,\ldots,u_K]$ dostarczanych sekwencyjnie do kodera, gdzie tworzone jest słowo kodowe $\mathbf{x}=[x_1,x_2,\ldots,x_N]$ o długości $N>K$ bitów, przesyłane nieidealnym kanałem telekomunikacyjnym. Kolejne słowa informacyjne $\mathbf{u}$ są przekształcane w słowa kodowe $\mathbf{x}$ niezależnie od siebie. Parametrem określającym stopień nadmiarowości kodu jest tzw. stopa kodu $R=K/N$.

Konkretny kod LDPC charakteryzowany jest przez macierz kontrolną kodu $\mathbf{H}_{M\times N}$ (gdzie $M=N-K$), która jest macierzą rzadką. Macierzowe równanie kontrolne (w ciele $GF(2)$):

$$\mathbf{H}\mathbf{x}^T=\mathbf{0}_{M\times 1} \tag{1}$$

pozwala sprawdzić, czy dany wektor $\mathbf{x}$ jest prawidłowym słowem kodowym. Jeśli równanie (1) nie jest spełnione, to w kanale wystąpiło przekłamanie, które należy skorygować.

W celu osiągnięcia niskiej bitowej stopy błędów systemu, kody definiowane są dla dużych bloków danych (typowo wiele tysięcy bitów), stąd macierz kontrolna ma również duży rozmiar. Dla osiągnięcia dobrych własności korekcyjnych kodu konieczne jest odpowiednie skonstruowanie macierzy. Nietrywialne jest również zadanie projektowania efektywnych modułów kodera i (w szczególności) dekodera sprzętowego, zapewniających odpowiednią przepustowość przy ograniczonych zasobach sprzętowych. Efektywna implementacja konfigurowalnego dekodera, która została opracowana, może być zastosowana dla pewnej podklasy kodów LDPC zorientowanych na implementację, nazywanych kodami AA-LDPC (Architecture Aware LDPC [3]).

W kolejnych sekcjach przedstawiona zostanie definicja kodu AA-LDPC, zaprezentowana zostanie pokrótce struktura opracowanego dekodera, możliwości jego konfiguracji oraz przedstawione będą wyniki implementacji modułów dekodera dla kilku wybranych kodów AA-LDPC.

2. Kody AA-LDPC

Pojęcie kody AA-LDPC [3] jest w ogólności używane do określenia podklasy kodów o macierzy kontrolnej $\mathbf{H}_{DP\times EP}$ składającej się z $D\times E$ podmacierzy kwadratowych $\mathbf{P}_{d,e}$, $d=1,\ldots,D$, $e=1,\ldots,E$:

$$\mathbf{H}=\begin{bmatrix}\mathbf{P}_{1,1} & \mathbf{P}_{1,2} & \cdots & \mathbf{P}_{1,E}\\ \mathbf{P}_{2,1} & \mathbf{P}_{2,2} & \cdots & \mathbf{P}_{2,E}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{P}_{D,1} & \mathbf{P}_{D,2} & \cdots & \mathbf{P}_{D,E}\end{bmatrix} \tag{2}$$

gdzie każda z podmacierzy $\mathbf{P}_{d,e}$ o rozmiarze $P \times P$ jest macierzą zerową lub też permutacją macierzy jednostkowej $\mathbf{P}_{d,e} = \mathbf{I}_{\theta(p)}$, tzn. macierzą kwadratową składającą się z wszystkich kolumn macierzy jednostkowej $\mathbf{I}_{P \times P}$ ułożonych w dowolnej kolejności definiowanej przez permutację $\theta(p)$, $p = 1,\ldots,P$. Permutacja macierzy jednostkowej posiada zatem dokładnie jedną jedynkę w każdej kolumnie i w każdym wierszu.

Macierz bazowa $\mathbf{W}$ kodu AA-LDPC to macierz o rozmiarze $D \times E$, w której element $w_{d,e} = 1$ wtedy i tylko wtedy gdy $\mathbf{P}_{d,e}$ jest permutacją macierzy jednostkowej; w przeciwnym razie $w_{d,e} = 0$.

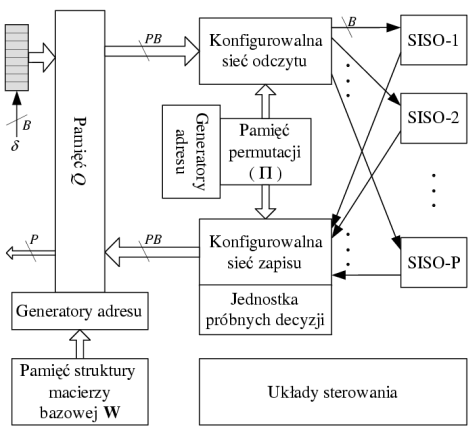
Jak się okazuje [3, 7], dla kodów o macierzy kontrolnej spełniającej powyższą definicję, architektura dekodera może być w postaci szeregowo-równoległego połączenia jednostek obliczeniowych realizujących iteracyjny algorytm dekodowania. Jest to jedyna architektura dekodera sprzętowego umożliwiająca uzyskanie odpowiedniej przepustowości dekodera przy ograniczonych zasobach sprzętowych, dla kodów o dużej długości bloku N.

3. Struktura dekodera kodów AA-LDPC

Skupienie uwagi na module dekodera wynika z faktu, że operacja dekodowania jest bardziej złożona koncepcyjnie i obliczeniowo niż operacja kodowania. Zadanie opracowania efektywnego dekodera sprzętowego jest znacznie trudniejsze niż kodera. Poza tym znanych jest wiele modyfikacji suboptymalnego algorytmu dekodowania. Każda konfiguracja dekodera może dawać nieco inne rezultaty (możliwości korekcyjne systemu koder-dekoder), jednocześnie wymagając innej liczby zasobów sprzętowych.

Wykorzystywane w praktyce algorytmy dekodowania należą do grupy iteracyjnych algorytmów przekazywania wiadomości (*Message Passing Algorithms*), obrazowanych przez tzw. graf Tannera [2, 3]. Jest to graf dwudzielny, zawierający wierzchołki bitowe (bity słowa kodowego) i kontrolne (równania kontrolne) skojarzone z pewnymi elementarnymi operacjami realizowanym w procesie dekodowania. Natomiast krawędzie grafu Tannera wskazują sposób przekazywania wiadomości pomiędzy wierzchołkami w kolejnych iteracjach. Wiadomości są funkcjami LLR (*Log-Likelyhood Ratio*) prawdopodobieństw wartości poszczególnych bitów, a metoda wyznaczania i propagacji tych wartości znana jest jako LLR-BP (LLR *Belief Propagation*). Opis iteracyjnych algorytmów dekodowania dostępny jest w bogatej literaturze, np. [2, 4], a odmiana zwana algorytmem TDMP (*Turbo-Decoding Message-Passing*) zastosowana w opracowanym dekoderze opisana jest m.in. w [5].

Struktura dekodera przedstawiona jest na rys. 1. Szczegółowy opis zastosowanego algorytmu oraz bloków funkcjonalnych dekodera przedstawiono w pracach [6, 7]. W niniejszej pracy opisane zostaną możliwości konfiguracji dekodera oraz wyniki implementacji.



Rys. 1. Struktura dekodera LDPC
Fig. 1. LDPC decoder structure

4. Możliwości konfiguracji dekodera

Podstawowymi elementami realizującymi iteracyjny algorytm są bloki obliczeniowe SISO (*Soft In Soft Out*), pamięć wiadomości Q oraz konfigurowalne sieci zapisu i odczytu danych z pamięci (zespoły multiplekserów). Dekoder został zaimplementowany w postaci syntezowalnego opisu w języku VHDL, przy czym dostosowanie dekodera do konkretnego kodu AA-LDPC wymaga odpowiedniej rekonfiguracji pewnych modułów dekodera (zmodyfikowania pewnych fragmentów opisu poddawanego syntezie), a mianowicie:

- Zaimplementowania odpowiedniej liczby jednostek obliczeniowych SISO, w zależności od rozmiaru podmacierzy macierzy kontrolnej kodu oraz wymagań co do przepustowości. Przepustowość dekodera wyznaczyć można z zależności:

$$TH \cong \frac{f_{clk} \cdot P}{\overline{d_C} \cdot i} \tag{3}$$

gdzie f_{clk} to częstotliwość sygnału zegarowego; P to liczba jednostek obliczeniowych SISO, w typowym przypadku równa rozmiarowi podmacierzy macierzy kontrolnej; $\overline{d_C}$ to średnia waga wiersza macierzy kontrolnej (średnia liczba niezerowych elementów w wierszu), natomiast i oznacza liczbę iteracji procesu dekodowania. Dla wyznaczenia średniej przepustowości dekodera należy do powyższego wzoru podstawić średnią liczbę wymaganych iteracji, która w typowym przypadku wynosi 6...10. W wynikach przedstawionych w niniejszej pracy do wyznaczenia przepustowości przyjęto najgorszy przypadek, tzn. $i = 10$.

- Zdefiniowania zawartości pamięci struktury macierzy bazowej $\mathbf{W}$. Przykładowy fragment definicji zawartości pamięci:

```
type WRam_type is array(0 to dsubmatr-1)
      of std_logic_vector(6 downto 0);
constant WRam_content: WRam_type
     := ("1000001", "0010010", "0011100",
         "0100100", "0110000", "0110110",
         "1000010", "0010111",...
                                   (itd.)
```

Jedynka na pierwszym bicie elementu tablicy `Wram_content` wskazuje „nowy wiersz”, a pozostałe 6 bitów wskazuje numer kolumny niezerowego elementu w „aktualnym” wierszu. Tak więc powyższy przykład odnosi się do macierzy $\mathbf{W}$, w której w pierwszym wierszu jedynki występują w kolumnach: 1, 18, 28, 36, 48, 54. Liczba komórek pamięci definiowana przez stałą `dsubmatr` równa jest sumarycznej liczbie jedynek w macierzy bazowej $\mathbf{W}$.

- Zaimplementowania konfigurowalnych sieci zapisu i odczytu (sieci multiplekserów) o odpowiedniej liczbie wejść / wyjść oraz zdefiniowania zawartości pamięci permutacji, której poszczególne elementy sterują siecią. Przykładowa definicja zawartości pamięci permutacji:

```
type PiRam_type is array(0 to dsubmatr-1)
    of std_logic_vector(0 to Pi_len*P-1);
   --tablica permutacji (Siso -> Pamięć Q)
constant PiRamWr_content: PiRam_type
    :=("11110000001000010011111010100110011
    11100010010101101111001101111100100",
    "00000001001000110100010101100111100011
    00110101011110011010111101111",...
                                   (itd.)
```

Sieć zapisu łączy B-bitowe wiadomości z P jednostek SISO w jedno PB-bitowe słowo zapisywane do pamięci. Kolejność wiadomości w słowie w kolejnych sub-iteracjach jest określana przez zawartość kolejnych komórek pamięci permutacji (odpowiadających kolejnym podmacierzom permutacji $\mathbf{P}_{d,e}$ macierzy kontrolnej kodu). W powyższym przykładzie, w pierwszej sub-iteracji w słowie zapisywanym do pamięci umieszczone są wia-

domości kolejno z następujących jednostek SISO: 15, 0, 2, 1, 3, 14, 10, 6, 7, 8, 9, 5, 11, 12, 13, 4.

- Pewnych modyfikacji układów sterowania (automatów określających tryby pracy jednostek obliczeniowych). Sposób działania układów sterowania zmienia się w zależności od liczby jedynek w kolejnych wierszach macierzy bazowej $\mathbf{W}$.

Możliwa jest również konfiguracja parametrów dekodera, decydujących o własnościach korekcyjnych systemu kodowania (w zależności od wymagań), tzn.:

- Długości stałoprzecinkowego słowa wiadomości B (z przedziału od 4 do 8 bitów), która ma wpływ na własności korekcyjne systemu. Eksperymentalnie ustalono, że znakomite własności uzyskuje się już dla $B = 6 \ldots 7$, tym niemniej możliwe jest również wykorzystanie mniejszych długości słowa dla uzyskania oszczędności zasobów kosztem pogorszenia własności korekcyjnych.
- Sposobu wyznaczania wiadomości (skojarzonego z pojedynczym wierzchołkiem kontrolnym grafu). Oprócz podstawowego algorytmu LLR-BP dostępne są jego uproszczone wersje, między innymi *Min-Sum*, *Corrected Min-Sum*, *Normalized Min-Sum* [3, 4, 8]. Do projektu włączany jest opis jednostki SISO działającej zgodnie z wybranym algorytmem.

Dostosowanie opisu dekodera jest wykonywane automatycznie za pomocą opracowanego oprogramowania. W celu integracji modułów projektowania dekodera z narzędziami zapewniającymi dużą swobodę implementacji algorytmów tworzenia macierzy kontrolnej, środowisko zostało oparte na oprogramowaniu Matlab. Do implementacji dekodera wykorzystano narzędzia firmy Xilinx oraz zestaw uruchomieniowy z układem rodziny VirtexII. Opracowana platforma programowo-sprzętowa służy do prac nad projektowaniem kodów AA-LDPC o dobrych własnościach korekcyjnych. Fizyczna implementacja dekodera w układzie FPGA pozwala na weryfikację skuteczności dekodera, jak również symulację systemów kodowania pozwalającą eksperymentalnie wyznaczać własności projektowanych kodów [6] w znacznie krótszym czasie niż z wykorzystaniem symulacji czysto programowych.

5. Wyniki implementacji

Przedstawione poniżej parametry dekodera implementowanego w FPGA pochodzą z raportów syntezy wykonywanej za pomocą programu XST z pakietu Xilinx ISE wersji 8.2, dla układu XC2V3000-4 rodziny VirtexII. W tabeli 1 przedstawiono wymagane zasoby strukturalne (liczbę komórek typu Slice i bloków pamięci) oraz uzyskaną przepustowość dekoderów kilku różnych kodów. Tabela 2 prezentuje zasoby wymagane przez poszczególne bloki funkcjonalne dekodera o P=24 jednostkach SISO.

Tab. 1. Parametry dekoderów (alg. *Corrected Min-Sum*, wiadomości 6-bitowe)
Tab. 1. Decoders parameters (Corrected Min-Sum Algorithm, 6b Messages)

Parametry kodu	f_{clk}	Przepust.	SLICEs	BRAMs
N=1024, R=0.5, P=16	95MHz	24.5Mb/s	2616 (18%)	8 (8%)
N=2640, R=0.5, P=24	95MHz	36.8Mb/s	4531(31%)	13 (14%)
N=2048, R=0.75, P=32	95MHz	70Mb/s	6511(45%)	17 (18%)

Tab. 2. Bloki funkcjonalne dekodera kodu o parametrach N=2640, R=0.5, P=24
Tab. 2. Decoder blocks for code with parameters N=2640, R=0.5, P=24

Blok	SLICEs	BRAMs
24 moduły obliczeniowe SISO	2520	0
Konfigurowalne sieci zapisu i odczytu	1794	0
Pamięć permutacji	0	8
Pamięć wiadomości Q	0	4
Pamięć struktury mac. bazowej $\mathbf{W}$	0	1
Jednostka próbnych decyzji	30	0
Układy sterowania i generatory adresów	187	0
RAZEM	**4531**	**13**

Zależność pomiędzy liczbą jednostek dekodera (P) a wymaganymi zasobami jest nieliniowa ze względu na nieliniowy wzrost powierzchni konfigurowalnych sieci zapisu i odczytu. Zmiana P powoduje powiem zmianę zarówno liczby wejść, jak i liczby wyjść sieci multiplekserów. Wydaje się zatem, że zastosowaniem opracowanej struktury dekodera powinny być przede wszystkim systemy o niskiej i średniej przepustowości (małej liczbie jednostek obliczeniowych dekodera). Tym niemniej uzyskana przepustowość rzędu 70Mb/s, przy wykorzystaniu około połowy zasobów układu XC2V3000 wydaje się być zadowalającym wynikiem (spełnia przykładowo wymagania transmisji sygnału telewizji cyfrowej), a główną zaletą opracowanego rozwiązania jest jego uniwersalność i możliwość rekonfiguracji dla dowolnego kodu z szerokiej klasy AA-LDPC, jak również wykorzystania szeregu różnych suboptymalnych algorytmów dekodowania, w zależności od potrzeb.

6. Podsumowanie

W artykule przedstawiono uniwersalny dekoder sprzętowy kodów AA-LDPC, implementowany w układzie FPGA oraz pokrótce omówiono możliwości jego konfiguracji. Szczególnie istotną wartością opracowanego środowiska jest możliwość szybkiego (automatycznego) dostosowania dekodera do dowolnego kodu AA-LDPC. Użytkownik może również dokonać wyboru algorytmu wyznaczania wiadomości oraz długości słowa. Modyfikacje struktury oraz konfiguracja dekodera przeprowadzana jest automatycznie za pomocą opracowanego oprogramowania w środowisku Matlab. Przedstawiono również wyniki implementacji dekodera dla kilku przykładowych kodów. Poprawność funkcjonowania dekoderów została zweryfikowana z wykorzystaniem zestawu ewaluacyjnego z układem XC2V3000 rodziny VirtexII. Dekoder jest elementem platformy programowo-sprzętowej wspierającej badania dotyczące projektowania „dobrych" kodów AA-LDPC, jak również może być natychmiast wykorzystany w praktycznym systemie transmisji z kodowaniem LDPC. Uzyskane wyniki pozwalają bowiem na implementację dekodera (spełniającego wymogi np. standardu transmisji cyfrowej sygnału telewizyjnego) w średniej wielkości układzie FPGA.

7. Literatura

[1] R. G. Gallager: Low-Density Parity-Check Codes. MIT Press, Cambridge, MA, 1963.
[2] D. J. C. MacKay: Good Error-Correcting Codes Based on Very Sparse Matrices. IEEE Transactions on Information Theory, vol. 45, no. 2, March 1999, pp. 399–431.
[3] M. M. Mansour, N. R. Shanbhag: High Throughput LDPC Decoders. IEEE Transactions on Very Large Scale Integration Systems, vol. 11, no. 6, December 2003, pp. 976–996.
[4] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, X.-Y. Hu: Reduced-Complexity Decoding of LDPC Codes. IEEE Transactions on Communications, vol. 53, no. 8, August 2005, pp. 1288–1299.
[5] M. M. Mansour: A Turbo-Decoding Message-Passing Algorithm for Sparse Parity-Check Matrix Codes. IEEE Transactions on Signal Processing, vol. 54, no. 11, November 2006, pp. 4376–4392.
[6] W. Sułek, D. Kania: Code Construction Algorithm for Architecture Aware LDPC Codes with Low-Error-Floor. IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering, SIBIRCON 2008, Novosibirsk 2008.
[7] W. Sułek: Implementacja modułu sprzętowego dekodera kodów AA-LDPC. Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjne, no. 8-9/2008, pp. 1229–1240.
[8] F. Zarkeshvari, A. H. Banihashemi: On Implementation of Min-Sum Algorithm for Decoding Low-Density Parity-Check (LDPC) Codes. IEEE Globecom, Taipei, Taiwan, August 2002, pp. 1349–1353.

Artykuł recenzowany