Mateusz KUC, Wojciech SUŁEK, Dariusz KANIA

Politechnika Śląska, Instytut Elektroniki

doi:10.15199/48.2019.14

Sprzętowa implementacja dekodera LDPC w strukturze FPGA*

Streszczenie. *W artykule przedstawiono sprzętową implementację dekodera LDPC (ang. Low-Density Parity-Check) w strukturze FPGA (ang. Field Programmable Gate Array). W celu zredukowania złożoności implementacji wykorzystano algorytm MIN-SUM dla węzłów bitowych (CNU) i węzłów kontrolnych (VNU). W zrealizowanym dekoderze wykorzystano kod regularny (3,6) macierzy kontrolnej o wymiarach 512 x 1024 i zaimplementowano 4-bitową magistralę danych. Poprawność działania dekodera zweryfikowano praktycznie.*

Abstract. *The article presents the hardware implementation of the LDPC decoder (Low-density parity-check) in the FPGA structure (Field Programmable Gate Array). In order to reduce the complexity of the implementation, the Min-Sum algorithm for bit nodes (CNUs) and control nodes (VNUs) was used. The presented implementation was created using a regular code (3.6) of a 512 x 1024 control matrix. A 4-bit data bus was implement.* ***(Hardware implementation of the LDPC decoder in the FPGA structure)***

Słowa kluczowe: kody LDPC, FPGA, Min-Sum, implementacja sprzętowa
Keywords: LDPC, FPGA, Min-Sum, hardware implementation

Wstęp

We współczesnym świecie duże znaczenie ma umiejętność poprawnego przesyłu danych w jak najkrótszym czasie. Przetwarzanych jest coraz więcej danych, a ich przesyłanie staje się coraz bardziej skomplikowane. W świecie fizycznym występują różnego typu zakłócenia, które wpływają na przesyłany sygnał. W tej sytuacji bardzo ważne staje się wykrywanie powstałych błędów. Znane są metody kodowania które pozwalają nie tylko wykrywać błędy transmisji, ale również potrafią je korygować. Kody te wymagają przesyłania dodatkowych bitów kontrolnych, co w pewnym stopniu ogranicza liczbę danych, możliwych do przesyłania w kanale o określonej pojemności. Niektóre kody mają bardzo korzystny stosunek przesyłanych danych do pojemności kanału. Przykładem takich kodów są kody LDPC [1] i Turbo kody [2].

Blokowe kody LDPC zostały opracowane przez R. G. Gallagera w 1962r. [1] Z powodu ograniczeń ówczesnych układów Gallager nie mógł przedstawić praktycznej implementacji. W tej sytuacji zapomniano o nich i dopiero D. J. C. MacKay przedstawił je ponownie w prasie naukowej w późniejszym czasie [3]. Obecnie kody te zdobywają coraz większą popularność dzięki możliwości efektywnej implementacji w układach ASIC (ang. Application-Specific Integrated Circuits) i FPGA. Klasyczny algorytm dekodowania, oparty na iteracyjnej propagacji wiadomości (ang. Belief Propagation - BP) pomiędzy węzłami realizującymi jednostkowe obliczenia, doczekał się wielu odmian dostosowanych do implementacji w układach cyfrowych, jak np. LLR-BP (ang. Log-Likelihood Ratio Belief Propagation) [4], Min-Sum [5], Normalized-Min-Sum [5], Offset-Min-Sum [5] i TDMP (ang. Turbo Decoding Message Passing) [6]. Kody LDPC charakteryzują się znakomitymi własnościami korekcyjnymi dla bardzo dużych bloków danych. Najistotniejszym obszarem ich zastosowania są standardy transmisji danych z dużą przepływnością. Przykładem takich standardów mogą być między innymi: 802.3an (Ethernet 10GBase-T) [7], 802.11n/ac/ax (WiFi) [8], 802.11ad(WiGig) [9], 802.16e (WiMAX) [10], DVB-S2 [11] / DVB-T2 [12] (telewizja cyfrowa), sieci 5G [5].

Dany kod LDPC jest definiowany przez macierz kontrolną $\mathbf{H}$, która jest macierzą rzadką i może zostać przedstawiona graficznie za pomocą grafu Tannera [13]. Dekodery LDPC charakteryzują się możliwością prostego zrównoleglenia obliczeń, dzięki czemu można implementować je w formie szeregowej, szeregowo-równoległej i równoległej. Najlepszym kompromisem między złożonością implementacji a przepustowością charakteryzuje się architektura szeregowo-równoległa. W ten sposób jest implementowana większość dekoderów.

Celem artykułu jest przedstawienie sprzętowej realizacji dekodera kodów LDPC w strukturze FPGA i porównanie jego parametrów z konkurencyjnymi rozwiązaniami znanymi z literatury.

W dalszej części artykułu przedstawiono podstawowe informacje na temat wykorzystania kodów LDPC, opracowanych standardów oraz podstaw teoretycznych algorytmu Min-Sum. Po szeroko rozumianym wprowadzeniu zaprezentowano sprzętową implementację dekodera LDPC wraz z wynikami syntezy oraz analizą badań symulacyjnych. Artykuł kończy podsumowanie wskazujące kierunki dalszych prac.

Podstawowe aspekty teoretyczno-implementacyjne

Istnieje wiele odmian algorytmów dekodowania kodów LDPC, wśród których największą popularnością w implementacjach sprzętowych cieszy się algorytm Min-Sum. Znanych jest szereg prac poświęconych licznym odmianom tego algorytmu dekodowania ([5], [14], [15], [16], [17]).W podstawowej formie algorytmu Min-Sum można wyróżnić następujące etapy procesu dekodowania:

1. *Inicjalizacja*: Przypisanie prawdopodobieństw LLR (ang. Log-Likelyhood Ratio), które są wartościami wejściowymi algorytmu dekodowania (dla wszystkich $n\varepsilon(1,M)$ i wszystkich $m\varepsilon M(n)$):

$$Q_{nm} := L(x_n|y_n) = \ln\left(\frac{P(x_n=0|y_n)}{P(x_n=1|y_n)}\right) \tag{1}$$

2. *Krok poziomy*: Wyznaczanie wartości minimalnych (dla wszystkich $m\varepsilon(1,N)$ i wszystkich $n\varepsilon N(m)$):

$$R_{mn} := \left[\prod_{k\varepsilon N(m)\backslash n} \operatorname{sgn}(Q_{km})\right] \min_{k\varepsilon N(m)\backslash n} |Q_{km}| \tag{2}$$

3. *Krok pionowy*: Sumowanie prawdopodobieństwa LLR i wartości minimalnych (dla wszystkich $n\varepsilon(1,M)$ i wszystkich $m\varepsilon M(n)$):

$$Q_{nm} := L(x_n|y_n) + \sum_{k\varepsilon M(n)\backslash m} R_{kn} \tag{3}$$

4. *Prawdopodobieństwa pseudo-posteriori*: Wyznaczanie prawdopodobieństw LLR (dla wszystkich n):

$$Q_n := L(x_n|y_n) + \sum_{k\varepsilon M(n)} R_{kn} \tag{4}$$

5. *Próbne decyzje*: Podejmowanie twardych decyzji (dla wszystkich $n\varepsilon(1, M)$):

$$\hat{x}_n := \begin{cases} 1 & \text{gdy } Q_n < 0 \\ 0 & \text{gdy } Q_n \geq 0 \end{cases} \tag{5}$$

6. *Sprawdzanie równań kontrolnych*:

$$\mathbf{H}\hat{x}^T = 0 \tag{6}$$

Gdzie:

Q_{nm} – wartość LLR z n-tego wierzchołka bitowego do m-tego wierzchołka kontrolnego,

$L(x_n|y_n)$ – LLR prawdopodobieństw *a priori* dla bitu n-tego,

$\mathbf{x} = [x_1, x_2, \ldots, x_n]$ – wektor kodowy,

$\mathbf{y} = [y_1, y_2, \ldots, y_n]$ – wektor odebrany,

R_{mn} – wartość LLR z m-tego wierzchołka kontrolnego do n-tego wierzchołka bitowego,

$N(m)$ – numery kolumn w macierzy kontrolnej $\mathbf{H}$ zawierających jedynkę w wierszu m-tym,

$M(n)$ – numery wierszy w macierzy kontrolnej $\mathbf{H}$ zawierających jedynkę w kolumnie n-tej,

$\hat{x}_n$ – wektor zdekodowany.

Dekoder działający zgodnie z powyższym algorytmem na początku jest inicjalizowany wartościami pochodzącymi z demodulatora w postaci logarytmicznego stosunku prawdopodobieństw (LLR) (1). W kroku poziomym (2) wykonywane są operacje wyznaczania wartości minimalnej z modułu liczb Qkm oraz iloczyn ich znaków. Wybór liczb uwzględnianych do obliczeń jest związany z zastosowaną macierzą kontrolną $\mathbf{H}$. W kroku pionowym (3) wykonywane jest sumowanie wcześniej wyznaczonych liczb (2) wraz z wartością *a priori* (1). Krok poziomy i pionowy odpowiada kolejno węzłom kontrolnym i bitowym w grafie Tannera [13]. Warto zauważyć, że wyznaczanie prawdopodobieństw pseudo-posteriori (4) można połączyć z obliczeniami wykonywanymi w kroku pionowym (3). Dzięki temu implementacja w układzie FPGA może zostać uproszczona. Ostatecznie są wyznaczane twarde decyzje (5) na podstawie których są sprawdzane równania kontrolne (6). Jeżeli równania kontrolne zostały spełnione (6) bądź osiągnięto maksymalną liczbę iteracji dekodera wynik zostaje przekazany do odbiorcy.

Algorym Min-Sum posiada dwie istotne odmiany: Normalized Min-Sum i Offset Min-Sum, które charakteryzują zależności (7) i (8) w kroku poziomym, przy pozostałych krokach takich samych jak w Min-Sum.

Normalized Min-Sum:

$$R_{mn} := \left[\prod_{k\varepsilon N(m)\backslash n} \operatorname{sgn}(Q_{km}) \right] \frac{\min\limits_{k\in N(m)\backslash n} |Q_{km}|}{\alpha} \tag{7}$$

$$0 < \alpha < 1$$

Offset Min-Sum:

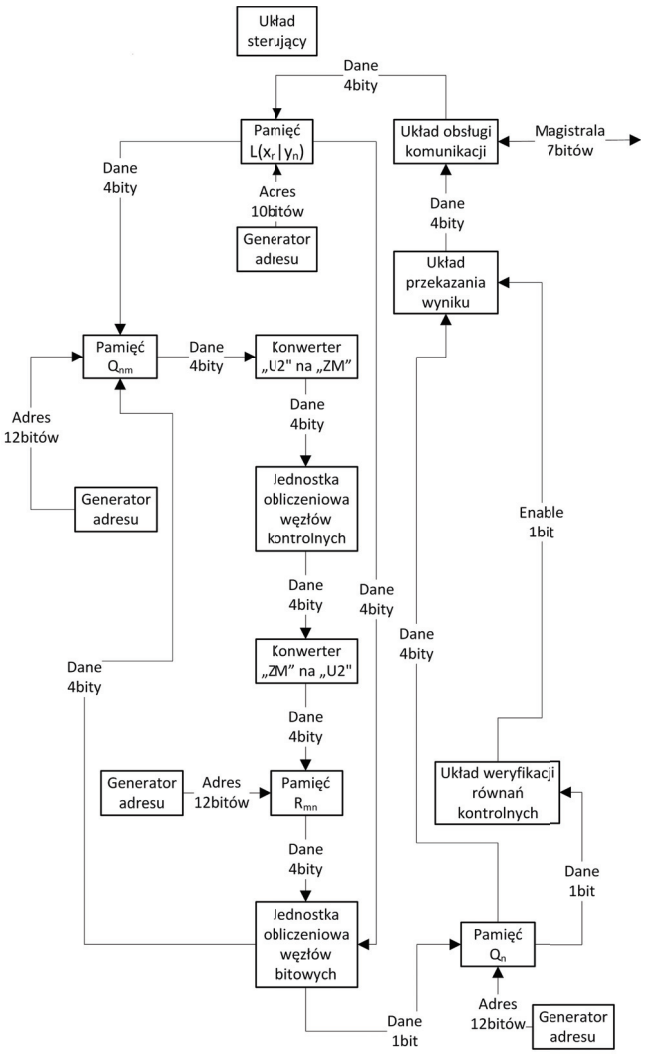
$$R_{mn} := \left[\prod_{k\varepsilon N(m)\backslash n} \operatorname{sgn}(Q_{km}) \right] \min_{k\in N(m)\backslash n} |Q_{km}| - \alpha \tag{8}$$

$$\alpha > 0$$

Jak można zauważyć, w zależnościach (7) i (8) pojawia się parametr α, który jest dobierany doświadczalnie. Jego wprowadzenie pozwala uzyskać lepsze wyniki dekodowania, ale nieco zwiększa złożoność implementacji dekodera w strukturze FPGA ([5], [15], [17]).

Implementacja dekodera LDPC

Centralną częścią zaimplementowanego dekodera LDPC są jednostki obliczeniowe realizujące krok poziomy (jednostka węzłów kontrolnych) i pionowy (jednostka węzłów bitowych). Dane pomiędzy jednostkami są przesyłane z wykorzystaniem pamięci wiadomości Q_{nm} i R_{mn}. Przesyłane dane są 4-bitowe, co stanowi kompromis pomiędzy złożonością a skutecznością dekodowania [18]. Dane wejściowe (wartości LLR) 4-bitowe są transmitowane do dekodera za pomocą 7-bitowej magistrali. Schemat blokowy dekodera LDPC przedstawiono na Rys. 1.

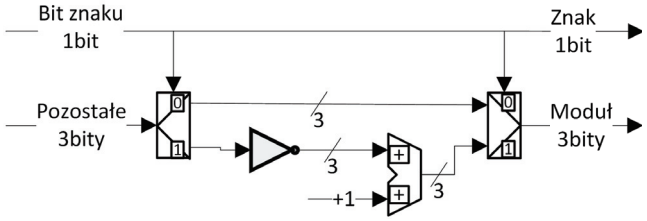


Rys. 1. Schemat blokowy dekodera LDPC w układzie FPGA

Układ obsługi komunikacji zapewnia wymianę danych pomiędzy komputerem a układem FPGA. Wykorzystuje 7-bitową magistrale, co umożliwia testowanie dekodera. Pamięć $L(x_n|y_n)$ przechowuje 1024 dane 4-bitowe. Adresy pamięci $L(x_n|y_n)$ są ustawiane w generatorze adresu.

W pierwszym etapie, gdy dane są odbierane z mikrokontrolera, generator adresu wskazuje kolejne komórki pamięci. Po zakończeniu procesu odbioru danych, rozpoczyna się inicjalizacja dekodera związana z przekazaniem danych dopamięci Q_{nm}, określona zależnością (1). Pamięć Q_{nm} przechowuje 3072 liczby 4 bitowe, gdyż jej rozmiarjest równy liczbie niezerowych elementów w $\mathbf{H}$. Po zakończeniu pro-

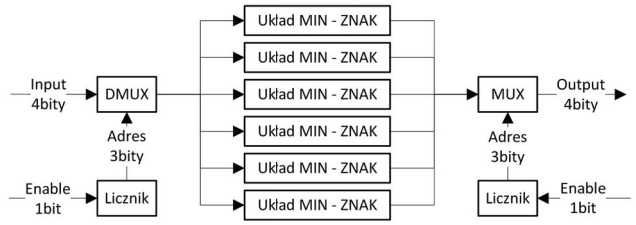
cesu inicjalizacji dane z pamięci Q_{nm} są przekazywane do konwertera „U2" do "ZM" (Znak-Moduł), którego schemat przedstawiono na rys. 2. Dane przekazywane są do konwertera w postaci 4-bitowej.



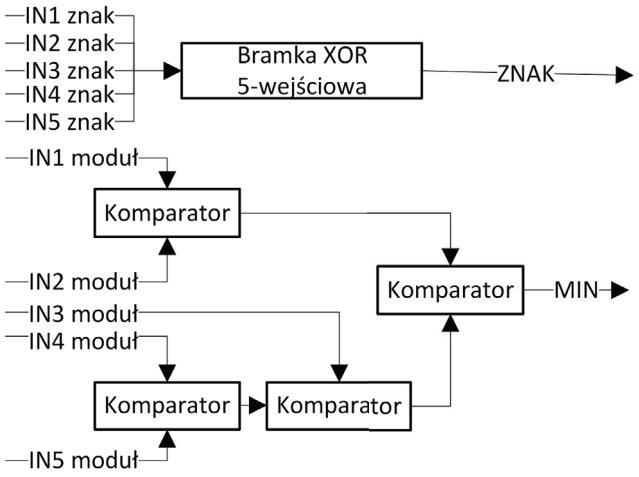
Rys. 2. Schemat konwertera U2 ↔ ZM

Konwerter przekształca liczby z kodu "U2" do "ZM" i odwrotnie. Zgodnie z zasadą konwersji bit znaku konwertowanej liczby decyduje, czy moduł liczby należy zanegować. W przypadku liczby ujemnej moduł liczby jest negowany i inkrementowany. Wynik konwersji przekazywany jest do jednostki obliczeniowej węzłów kontrolnych.

Jednostka obliczeniowa węzłów kontrolnych, przetwarzająca dane zgodnie z równaniem (2) przedstawiona jest na Rys. 3. Przetwarzane są kolejne wiadomości skojarzone z niezerowymi elementami w danym wierszu macierzy kontrolnej $\mathbf{H}$. Dane przekazywane są poprzez odpowiednio adresowany demultiplekser, który kieruje dane do rejestrów. Po skompletowaniu wszystkich danych rozpoczyna się proces wyznaczania wartości minimalnej i iloczynu znaków. Każdy układ MIN - ZNAK (obliczający wartość minimalną i znak) otrzymuje $T-1$ danych wejściowych, gdzie T oznacza liczbę niezerowych elementów w wierszu $\mathbf{H}$. Schemat układu obliczenia wartości minimalnej i znaku przedstawiono na Rys. 4.



Rys. 3. Jednostka obliczeniowa węzłów kontrolnych

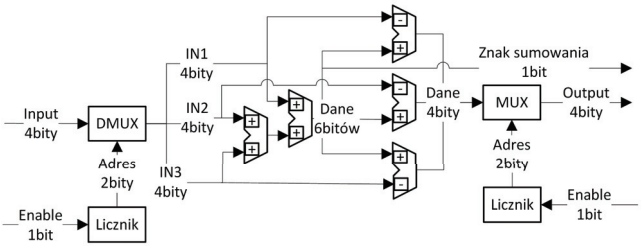


Rys. 4. Układ obliczania wartości minimalnej i znaku

W procesie obliczeń wykorzystano funktor XOR oraz zestaw komparatorów, które wyznaczają wartość minimalną. W wyniku uzyskiwana jest minimalna wartość modułu oraz znak, które po konwersji do postaci "U2" przekazywane są do pamięci R_{mn}.

Wyniki pochodzące z jednostki obliczeniowej węzłów kontrolnych są zapisywane do pamięci R_{mn}. Podobnie jak pamięć Qnm pamięć R_{mn} jest adresowana przez generator adresu i przechowuje 3072 liczby 4-bitowe. Po zapisaniu wszystkich otrzymanych wyników, dane są przekazywanie do jednostki obliczeniowej węzłów bitowych.

Jednostka obliczeniowa węzłów bitowych pobiera dane zarówno z pamięci R_{mn} jak i pamięci $L(x_n|y_n)$ otrzymując w ten sposób wszystkie wymagane dane do wykonania obliczeń opisanych równaniami (3) i (4). Schemat jednostki obliczeniowej węzłów bitowych przedstawiono na Rys. 5.



Rys. 5. Schemat jednostki obliczeniowej węzłów bitowych

Jednostka obliczeniowa węzłów bitowych pobiera trzy dane 4-bitowe z pamięci R_{mn} i jedną 4-bitową z pamięci $L(x_n|y_n)$. Następnie dane te są sumowane z uwzględnieniem ich znaków. W celu uniknięcia błędów zaokrągleń otrzymany wynik sumowania jest 6-bitowy. Następnie wynik ten zostaje przekazany do subtraktorów, które wykonują operację różnicy między wynikiem sumowania, a każdą daną przekazaną z pamięci R_{mn}. Otrzymane w ten sposób wyniki, zgodne z równaniem (3), przekazywane są na wejście multipleksera, który umożliwia przygotowanie danych do obliczeń wykonywanych w kolejnej iteracji. Bit znaku operacji sumowania odpowiada wynikowi opisanego równaniem (4) i jest przekazywany do pamięci Q_n.

Pamięć Q_n przechowuje 3072 bity znaków operacji sumowania jednostki obliczeniowej węzłów kontrolnych. Po zapisaniu wszystkich liczb generator adresu pamięci Q_n przekazuje kolejne pozycje węzłów kontrolnych do układu weryfikacji równań kontrolnych. Układ weryfikacji równań sprawdza parzystość jedynek każdego wiersza zgodnie z równaniem (5). Jest to wykonywane za pomocą funktora XOR. Jeżeli wszystkie równania kontrolne zostały spełnione (stwierdzono parzystość jedynek w każdej kolumnie) układ weryfikacji ustawia sygnał odblokowujący układ przekazywania wyniku.

Układ przekazywania wyniku jest buforem, który pozwala na transmisję danych z pamięci Q_n do układu obsługi komunikacji w momencie otrzymania sygnału odblokowującego pochodzącego z układu weryfikacji równań kontrolnych. Po zakończeniu obliczeń, wynik dekodowania znajduje się w pamięci Q_n. Obliczenia są kończone gdy wszystkie równania kontrolne zostały spełnione lub osiągnięto założoną liczbę iteracji dekodowania.

Środowisko testowe

Sprzętowe implementacje dekodera LDPC są związane z szeregiem uproszczeń, których istota sprowadza się do zmniejszenia złożoności obliczeniowej implementowanych algorytmów. Stosuje się szereg elementów wpływających na zmniejszenie precyzji obliczeń, które wpływają na architekturę dekodera w układzie FPGA. Punktem odniesienia, do którego porównuje się parametry sprzętowego dekodera jest zwykle programowy sposób dekodowania kodów LDPC, realizowany przez program komputerowy. W tym celu zwykle

implementuje się precyzyjny algorytm dekodowania LLR-BP. Bardzo często w badaniach symulacyjnych wykorzystuje się środowisko MATLAB [14].

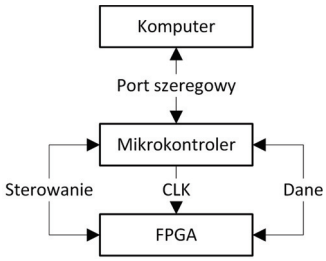
Podczas badań symulacyjnych wygenerowane wektory danych można z powodzeniem wykorzystywać do weryfikacji praktycznej dekodera realizowanego w strukturze FPGA. Uzyskane w ten sposób wyniki można porównać z dekoderem programowym. Do celów symulacji należy wybrać modulację i model kanału transmisyjnego. Typowo wykorzystywanym modelem kanału ([5], [16]) jest najprostszy model AWGN (ang. Additive White Gaussian Noise). Spośród podstawowych typów modulacji:

- BPSK (ang. Binary Phase Shift Keying) [14], [16],
- QPSK (ang. Quadrature Phase Shift Keying) [5],
- QAM (ang. Quadrature Amplitude Modulation) [16],

wybrano najprostsze rozwiązanie, jakim jest modulacja BPSK.

Przed przystąpieniem do badania działania dekodera LDPC opracowano w języku Python program komputerowy służący do wytwarzania danych potrzebnych podczas testowania dekodera. Zaimplementowano tam algorytm LLR-BP, model kanału AWGN i modulację BPSK. Opracowano graficzny interfejs (w języku C#) umożliwiający wczytywanie wektorów zakodowanych i zakłóconych, który pozwolił zautomatyzować proces testowania układu. Wektory zakłócone wysyłano poprzez port szeregowy komputera, a następnie wynik dekodowania odczytywano i porównywano z danymi pierwotnymi. Pozwoliło to na sprawne określenie parametrów dekodera takich jak: poziom bitowej stopy błędu BER (ang. Bit Error Rate) i blokowej stopy błędu FER (ang. Frame Error Rate).

Zdecydowano się na rozwiązanie, w którym dane z komputera były przesyłane do układu FPGA za pośrednictwem mikrokontrolera, komunikującego się z komputerem z wykorzystaniem typowej transmisji szeregowej UART (8 bitów). Na Rys. 6 przedstawiono schematycznie wymianę danych pomiędzy komputerem, a układem FPGA.



Rys. 6. Sposób komunikacji komputer – dekoder

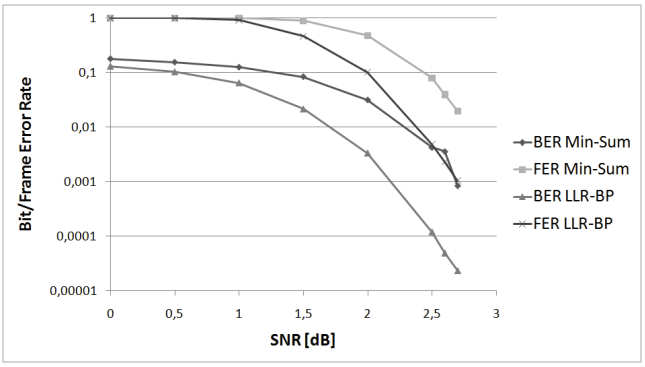
Sygnał CLK służy do synchronizacji mikrokontrolera i dekodera LDPC zrealizowanego w strukturze FPGA. Dwubitowy sygnał "Sterowanie" zapewnia współpracę pomiędzy układami ustalając jeden z czterech trybów pracy:

- mikrokontroler oczekuje na dane z komputera,
- mikrokontroler transmituje dane do układu FPGA,
- dekoder w układzie FPGA przetwarza dane,
- wynik dekodowania jest transmitowany do mikrokontrolera.

Wyniki symulacji i testów

Działanie dekodera LDPC wraz z przygotowanym środowiskiem testowym zweryfikowano praktycznie. Otrzymane wyniki symulacji porównano z wynikami dekodowania programowego. Wynik symulacji przedstawiono na Rys. 7.

Zgodnie z oczekiwaniami, algorytm LLR-BP wykazuje znacznie lepsze parametry dekodowania niż algorytm Min Sum, który zaimplementowano w układzie FPGA. Wyniki



Rys. 7. Wykres BER i FER dla symulacji i zaimplementowanego dekodera LDPC

wskazują na zmniejszenie zysku kodowania o ok. 0,5 dB dla dekodera sprzętowego (algorytm Min-Sum) w stosunku do pełnego algorytmu LLR-BP operującego na liczbach zmiennoprzecinkowych.

Poza osiągniętymi parametrami dekodowania należy zwrócić uwagę na wykorzystane zasoby sprzętowe w strukturze FPGA. W tabeli 1 przedstawiono zasoby sprzętowe dekodera z podziałem na elementy logiczne i pamięć. Przedstawiono procentowe wykorzystanie zasobów układu FPGA Cyclone IV EP4CE22F17C6N, w którym zaimplementowano dekoder. Zaprezentowane wyniki dotyczą dekodera LDPC (3,6) i macierzy kontrolnej o rozmiarze 512 x 1024. Sumarycznie wykorzystano 740 (3,32%) elementów logicznych i 105472 (17,34%) bitów pamięci.

Tab. 1. Wykorzystanie zasobów sprzętowych przez dekoder

	Elementy logiczne	%	Bity pamięci	%
Układ komunikacji i pamięci $L(x_n \mid y_n)$	74	0.331541	4096	0.673401
Inicjalizacja	41	0.183692	0	0
Pamięć Q_{nm}	88	0.394265	12288	2.020202
Krok poziomy	153	0.685484	0	0
Pamięć R_{mn}	45	0.201613	12288	2.020202
Krok pionowy	79	0.353943	0	0
Układ próbnych decyzji i pamięci Q_n	16	0.071685	3072	0.505051
Sprawdzanie równań kontrolnych i przekazywanie wyniku	47	0.210573	0	0
Sterowanie dekodera	126	0.564516	0	0
Generatory adresu	71	0.3181	73728	12.12121

Najwięcej zasobów sprzętowych scharakteryzowanych liczbą bitów pamięci (12%) wykorzystują generatory adresu. Elementy logiczne potrzebne do realizacji kroku poziomego wykorzystują z kolei największą liczbę elementów ogólnego przeznaczenia (0,69% dostępnych zasobów).

Podsumowanie

W artykule przedstawiono sprzętowe rozwiązanie dekodera kodów LDPC, które wraz z opracowaną platformą uruchomieniową będzie służyło do dalszych prac badawczo-rozwojowych, których celem jest opracowanie różnego typu implementacji energooszczędnych. Zaprezentowana w artykule platforma uruchomieniowa jest uniwersalna, a po drobnych modyfikacjach stwarza możliwość testowania dekoderów z innymi regularnymi macierzami kontrolnymi $\mathbf{H}$ (3,6).

** Praca częściowo finansowana ze środków Ministerstwa Nauki i Szkolnictwa Wyższego.*

***Autorzy**: mgr inż. Mateusz Kuc, Instytut Elektroniki, ul. Akademicka 16, 44-100 Gliwice, e-mail:matekuc827@student.polsl.pl, dr inż. Wojciech Sułek, Instytut Elektroniki ul. Akademicka 16, 44-100*

Gliwice, e-mail: Wojciech.Sulek@polsl.pl, prof. dr hab. inż. Dariusz Kania, Instytut Elektroniki, ul. Akademicka 16, 44-100 Gliwice, e-mail: dkania@polsl.pl

LITERATURA

[1] Gallager R. G.: Low-Density Parity-Check Codes, *MIT Press*, (1963)

[2] Berrou C., Glavieux A., Thitimajshima P.: Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes, *IEEE International Limit Conference on Communications*, 2 (1993), 1064–1070

[3] MacKay D. J. C.: Good Error-Correcting Codes Based on Very Sparse Matrices, *EEE Transactions on Information Theory*, 45 (1999), n. 2, 399 – 431

[4] Zengyou S., Jin Z., Juan D.: Research of LDPC decoding based on LLR BP algorithm, *Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*, 2 (2011), 889 – 892

[5] Li H., Guo J., Guo C., Wang D.: A low-complexity min-sum decoding algorithm for LDPC codes, *IEEE 17th International Conference on Communication Technology (ICCT)*, (2017), 102 – 105

[6] Zhao M., Zhang X., Zhao L., Lee C.: Design of a High-Throughput QC-LDPC Decoder With TDMP Scheduling, *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62 (2015), n. 1, 56 – 60

[7] Cohen A. E., Parhi K. K. : A Low-Complexity Hybrid LDPC Code Encoder for IEEE 802.3an (10GBase-T) Ethernet, *IEEE Transactions on Signal Processing*, 57 (2009), n. 10, 4085 – 4094

[8] Usman S., Mansour M. M., Chehab A.: A Multi-Gbps Fully Pipelined Layered Decoder for IEEE 802.11n/ac/ax LDPC Codes, *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, (2017), 194 – 199

[9] Li M., Lee Y., Huang Y., Liesbet Van der Perre: Area and energy efficient 802.11ad LDPC decoding processor, *Electronics Letters*, 51 (2015), n. 4, 339 – 341

[10] Adiono T., Prasetiadi A., Salbiyono A.: Efficient encoding for hardware implementation of IRA LDPC on 802.16 standard, *International Symposium on Intelligent Signal Processing and Communication Systems*, (2010), 1 – 4

[11] Hao H., Chen J., Zhou Y.: An irregular row weight problem resolution for DVB-S2 LDPC short frame, *IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, (2017), 45 – 48

[12] Newagy F. A., Elramly S. H. : Novel technique for scaling down LDPC code lengths in DVB-T2 standard, *International Conference on Telecommunications and Multimedia (TEMU)*, (2012), 180 – 184

[13] Tanner R. M.: A recursive Approach to Low Complexity Codes, *IEEE Transactions of Information Theory*, 27 (1981), n. 5, 533–547

[14] Sreemohan P. V., Nelsa S.: FPGA implementation of min-sum algorithm for LDPC decoder, *International Conference on Trends in Electronics and Informatics (ICEI)*, (2017), 821 – 826

[15] Roberts M. K., Sunny E.: Investigations on performance analysis of various soft decision based LDPC decoding algorithms, *International Conference on Inventive Computing and Informatics (ICICI)*, (2017), 175 – 179

[16] Xu Y., Szczecinski L., Rong B., Labeau F., He D., Wu Y., Zhang W.: Variable LLR Scaling in Min-Sum Decoding for Irregular LDPC Codes, *IEEE Transactions on Broadcasting*, 60 (2014), n. 4, 606 – 613

[17] Li W., Lin J., Wang Z., Banihashemi A. H.: An efficient post processing scheme to lower the error floor of LDPC decoders, *IEEE 17th International Conference on Communication Technology (ICCT)*, 2017, 122 – 126

[18] Zhao J., Zarkeshvari F.: On Implementation of Min-Sum Algorithm and Its Modifications for Decoding Low-Density Parity-Check (LDPC) Codes, *IEEE Transactions On Communications*, 53 (2005), n. 4, 549 – 554