POLITECHNIKA GDAŃSKA
WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko studenta: Michał Iwanicki
Nr albumu: 143234
Poziom kształcenia: Studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Profil: Algorytmy i modelowanie systemów

Imię i nazwisko studenta: Mateusz Bauer
Nr albumu: 184329
Poziom kształcenia: Studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Profil: Teleinformatyka

Imię i nazwisko studenta: Marcin Garnowski
Nr albumu: 172578
Poziom kształcenia: Studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Informatyka
Profil: Inteligentne systemy interaktywne

PRACA DYPLOMOWA INŻYNIERSKA

Tytuł pracy w języku polskim: Dydaktyczny symulator wybranych rozwiązań warstwy fizycznej sieci Ethernet

Tytuł pracy w języku angielskim: Didactic simulator of selected Ethernet physical layer solutions

Opiekun pracy: dr inż. Krzysztof Nowicki

Spis treści

Streszczenie

Celem pracy jest stworzenie symulatora wybranych rozwiązań warstwy fizycznej sieci Ethernet oraz przeprowadzenie zajęc laboratoryjnych z grupą studentów. Wybrane rozwiązania to: kodowanie korekcyjne Reeda-Solomona oraz modulacja amplitudy impulsów (PAM). Dodatkowo zakres prac ograniczono do skrętki oraz konkretnego standardu - 40GBASE-T. Pierwszy rozdział wprowadza do kodowania korekcyjnego, przedstawiając wymaganą teorię matematyczną, jego właściwości oraz zastosowania w standardach Ethernet. Następnie przedstawiono potrzebę wykorzystania modulacji i różne jego rodzaje. Dla każdej modulacji wymienione są zalety i wady oraz wykorzystanie w standardach Ethernet. W kolejnym rozdziale obszernie opisany jest standard 40GBASE-T. Przybliża między innymi: możliwości tego standardu oraz techniki i rozwiązania, które są w nim wykorzystane, . Następna część dotyczy stworzonego symulatora. Zawiera przegląd dostępnych narzędzi oraz tych, które rzeczywiście wybrano, w tym: język programowania i biblioteki, Ostatnią częścią jest sprawozdanie z prowadzonego laboratorium. Opisuje zarówno przebieg samych zajęć dydaktycznych, jak i przygotowanie sali i sprzętu. Do pracy zostały załączone dodatki: instrukcja laboratoryjna, kod symulatora oraz instrukcja instalacji wraz z wymaganymi plikami na dołączonym urządzeniu przenośnym.

Słowa kluczowe: Ethernet, symulacja, laboratorium, skrętka, kodowanie korekcyjne Reeda-Solomona, PAM, Python.

Abstract

The purpose of the work is to create a simulator of selected solutions of the physical layer of the Ethernet network and conduct laboratory classes with a group of students. The selected solutions are: Reed-Solomon error-correcting code and pulse-amplitude modulation (PAM). In addition, the scope of work was limited to twisted-pair and specific standard - 40GBASE-T. The first chapter introduces error-correcting coding, providing the required mathematical theory, its properties and applications in Ethernet standards. Afterwards, the need to use modulation and its different types are presented. For each modulation, the advantages, disadvantages and use in Ethernet standards are listed. The next chapter extensively describes the 40GBASE-T standard. It introduces, among other things: the capabilities of this standard and the techniques as well as solutions that are used in it. The next section deals with the simulator. It provides an overview of the available tools and those that were actually chosen, including: programming language and libraries, The last part is a report on the conducted laboratory classes. It describes both the course of the study itself and the preparation of the class room and equipment. The following appendices have been attached to the work: a laboratory instruction, simulator code and installation guide with the required files on the included data storage device.

Keywords: Ethernet, simulation, laboratory, twisted pair, Reed-Solomon error-correcting coding, PAM, Python.

1 Wstęp

1.1 Wprowadzenie

Celem projektu jest opracowanie programu umożliwiającego przeprowadzenie symulacji wybranych rozwiązań warstwy fizycznej Ethernet. Przyjęty, wraz z promotorem, zakres prac zakłada realizację następujących elementów:

- symulację sygnału w skrętce z możliwością modyfikacji parametrów wejściowych oraz kanału,
- symulację wybranch modulacji PAM,
- symulację kodowania korekcyjnego Reeda-Solomona.

Ponadto w ramach pracy inżynierskiej dyplomanci przygotują scenariusz zajęć laboratoryjnych wykorzystujących opracowany symulator, jak również przeprowadzą te zajęcia z grupą studentów. Scenariusz zawrze takie elementy, jak wstęp teoretyczny do omawianych zagadnień, zadania na wejściówkę, zadania do realizacji na laboratoriach oraz opracowanie wymienionych zadań.

Podczas przygotowywania dyplomu autorzy zapoznają się ze standardem IEEE 802.3, specyfiką oraz problemami występującymi w warstwie fizycznej w sieciach Ethernet. Dzięki realizacji tych zadań dyplomanci poszerzą swoją wiedzę w wymienionych obszarach oraz podzielą się wynikami swojej pracy ze studentami.

1.2 Autorstwo fragmentów pracy

- Michał Iwanicki
  - Kodowanie Reeda-Solomona — rozdział 4, wstęp teoretyczny dla studentów wraz z pytaniami na wejściówkę i zadaniami, implementacja w kodzie, implementacja interfejsu graficznego
  - Symulacja sygnału — implementacja symulacji w kodzie, częściowe autorstwo interfejsu.
- Marcin Garnowski
  - Symulator — rozdział 7,
  - Zajęcia dydaktyczne — sprawozdanie — rozdział 8, częściowe autorstwo,
  - Opis symulatora w instrukcji,
  - Pytanie na wejściówkę dotyczące symulatora,
  - Dodatki C, D, E,
  - Częściowe autorstwo interfejsu.

- Mateusz Bauer
  - Zajęcia dydaktyczne — sprawozdanie — rozdział 8, częściowe autorstwo
  - 40GBASE-T, rozdział 6
  - Modulacja, rozdział 5, wstęp teoretyczny do zajęć, pytania na wejściówkę, zadania laboratoryjne, implementacja interfejsu graficznego oraz implementacja modulacji w kodzie

2 Kodowanie korekcyjne Reeda-Solomona

2.1 Wstęp

Kodowanie korekcyjne Reeda-Solomona zostało stworzone przez Irvina S. Reeda oraz Gustava Solomona w 1960 roku [1]

Kody Reeda-Solomona charakteryzują się kilkoma parametrami [2]:

- alfabetem w ciele skończonym $\mathbb{F}_{p^m}$, $m > 1$, p jest liczbą pierwszą
- długością wiadomości k do zakodowania $k < 2^m$
- długością słowa kodowego n gdzie $k < n < 2^m$
- wielomianem generującym $g(x)$ dla kodów BCH

W kodach Reeda-Solomona wykorzystywanych w Ethernecie wykorzystujemy ciało $\mathbb{F}_2$ oraz ciała rozszerzone $\mathbb{F}_{2^m}, m \in \{2, 3, \ldots\}$.

2.2 Ciało skończone $\mathbb{F}_q$

Aby zrozumieć działanie kodu Reeda-Solomona, trzeba najpierw zrozumieć, czym jest ciało skończone $\mathbb{F}_q$ zwane też ciałem Galois GF(q). Ciało to jest ciałem K rzędu q, czyli takie, które zawiera jedynie q elementów. Aby struktura algebraiczna była ciałem, musi definiować 2 operacje zwane dodawaniem i mnożeniem. Te operacje muszą spełniać kilka warunków:

$a + (b + c) = (a + b) + c$	$\forall a, b, c \in K$	Łączność dodawania	(1)
$a \cdot (b \cdot c) = (a \cdot b) \cdot c$	$\forall a, b, c \in K$	Łączność mnożenia	(2)
$a + b = b + a$	$\forall a, b \in K$	Przemienność dodawania	(3)
$a \cdot b = b \cdot a$	$\forall a, b \in K$	Przemienność mnożenia	(4)
$a + 0 = a$	$\forall a \in K$	Element neutralny (0) dodawania	(5)
$a \cdot 1 = a$	$\forall a \in K$	Element neutralny (1) mnożenia	(6)
$a + (-a) = 0$	$\forall a \in K$	Element odwrotny (-a) dodawania	(7)
$a \cdot a^{-1} = 1$	$\forall a \in K \setminus \{0\}$	Element odwrotny (a^{-1}) mnożenia	(8)
$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$	$\forall a, b, c \in K$	Rozdzielność mnożenia względem dodawania	(9)

Aby stworzyć ciało rzędu p gdzie p jest liczbą pierwszą, można wykorzystać pierścień klas reszt $\mathbb{Z}/p\mathbb{Z}$ z elementami (10) i działaniem dodawania (11) i mnożenia (12)

$$\mathbb{Z}/p\mathbb{Z} = \{[a]_p \mid a \in \mathbb{Z}\} = \{[0]_p, [1]_p, [2]_p, \ldots, [p-1]_p\} \tag{10}$$

$$[a]_p + [b]_p = [a+b]_p \tag{11}$$

$$[a]_p \cdot [b]_p = [a \cdot b]_p \tag{12}$$

Dla p niebędących liczbami pierwszymi $\mathbb{Z}/p\mathbb{Z}$ nie będzie ciałem skończonym, ponieważ nie wszystkie elementy będą spełniały warunek (8). Dla $\mathbb{F}_2$ działania + i $*$ są równoważne operacjom logicznym XOR oraz AND zdefiniowanymi w tablicy 1

Tabela 1: Dodawanie i mnożenie w $\mathbb{F}_2$

a	b	+	·
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2.3 Rozszerzone ciało skończone $\mathbb{F}_{2^m}$

Aby stworzyć ciało skończone o rzędzie 2^m, $k \in \{1, 2, 3, \ldots\}$, musimy najpierw znaleźć nierozkładalny wielomian $p(x)$ stopnia k o współczynnikach $c_n \in \mathbb{F}_2$. Elementami tego ciała będą wielomiany o postaci $c_0 + c_1\alpha + c_2\alpha^2 + \cdots + c_{k-1}\alpha^{k-1}$, $c_n \in \{0, 1\}$. Zbiór tych elementów można zapisać jako zbiór wielomianów (13), zbiór k-krotek lub wartości binarnych zawierających współczynniki wielomianu (14).

$$\left\{ \sum_{n=0}^{k-1} c_n \alpha^n \,|\, c_n \in \{0,1\} \text{ for } 1 \le n \le k \right\} = \{0, 1, \alpha, 1+\alpha, \alpha^2, \ldots, 1+\alpha+\alpha^2+\cdots+\alpha^{k-1}\} \tag{13}$$

$$\{0,1\}^k = \{(0,0,0,\ldots,0), (1,0,0,\ldots,0), (0,1,0,\ldots,0), (1,1,0,\ldots,0), \ldots, (1,1,1,\ldots,1)\} \tag{14}$$

2.4 Wykorzystanie w standardach Ethernetowych

Różne kody Reeda-Solomona są wykorzystywane w wielu standardach Ethernet, wyróżnione w tablicy 2. Zapisane są one w formacie RS(n,k), gdzie n to długość słowa kodowego, a k to długość wiadomości. Na czerwono zaznaczono standard, na którym skupiono się w tej pracy.

2.5 Właściwości kodu

Kody Reeda-Solomona cechują się możliwością korekty $\lfloor \frac{n-k}{2} \rfloor$ oraz wykrycia $n-k$ błędnych symboli. Symbol w ciele $\mathbb{F}_{2^m}$ składa się z m bitów, co w przypadku błędów grupowych daje możliwość korekty maksymalnie $m \cdot \lfloor \frac{n-k}{2} \rfloor$ bitów bądź detekcji $m(n-k)$ przekłamanych bitów

Tabela 2: Kodowania RS w różnych standardach [3]

Kodowanie RS(n,k)	Standardy		
RS(528,514)	10GBASE-R,	25GBASE-R,	100GBASE-CR4,
	100GBASE-KR4,	100GBASE-SR4	
RS(544,514)	50GBASE-R,	100GBASE-KP4,	100GBASE-CR2,
	100GBASE-SR2,	100GBASE-DR,	100GBASE-FR1,
	100GBASE-LR1,	200GBASE-R,	400GBASE-R
RS(450,406)	1000BASE-T1		
RS(192,186)	25GBASE-T,	40GBASE-T	
RS(360,326)	2.5GBASE-T1,	5GBASE-T1,	10GBASE-T1

2.6 Tworzenie kodu

Istnieje wiele różnych sposobów tworzenia kodu, które tworzą kod o innych właściwościach.

2.6.1 Oryginalny sposób

Sposób kodowania przedstawiony w pracy Reeda i Solomona polega na stworzeniu wielomianu $p_m(x) = \sum_{i=0}^{k-1} m_i x^i$, gdzie $m_i \in \mathbb{F}_q$ to i-ty element wiadomości, po czym, za pomocą tego wielomianu, obliczane jest słowo kodowe $c(m) = (p_m(a_0), p_m(a_1), \ldots, p_m(a_{n-1}))$, gdzie a_i to różne elementy ciała $\mathbb{F}_q$.

Przykład:
Kod Reeda-Solomona dla $k = 7$, $n = 4$, $\mathbb{F}_{2^3}$, wielomian nierozkładalny $p(x) = x^3 + x + 1$ i wiadomości $m = 7654$ wynosi $c = 4022233$

$$p_m(x) = 7x^3 + 6x^2 + 5x + 4$$
$$c(m) = (p_m(0), p_m(1), p_m(2), p_m(3), p_m(4), p_m(5), p_m(6), p_m(7))$$
$$c(m) = (4, 0, 2, 2, 2, 3, 3)$$

2.6.2 Kod systematyczny

Za pomocą niewielkiej modyfikacji można stworzyć kod systematyczny, czyli taki, w którym słowo kodowe zawiera w sobie kodowaną wiadomość. Żeby stworzyć kod systematyczny, musimy zmodyfikować sposób tworzenia wielomianu w taki sposób by $p_m(x_i) = m_i$ dla $i \in \{0, 1, \ldots, k-1\}$.

Jednym ze sposobów stworzenia takiego wielomianu jest użycie metody interpolacji wielomianów. Słowo kodowe wygenerowane z tego wielomianu będzie zawierało wiadomość w pierwszych k

elementach.

$$\begin{aligned} C(m) &= (p_m(a_0), p_m(a_1), \ldots, p_m(a_{n-1})) \\ &= (m_0, m_1, \ldots, m_{k-1}, p_m(a_k), p_m(a_{k+1}), \ldots, p_m(a_{n-1})) \end{aligned}$$

Przykład:
Używając danych z poprzedniego przykładu z sekcji (2.6.1), obliczając wielomian $p_m(x)$ oraz używając interpolacji Lagrange, otrzymujemy

$$\begin{aligned} p_m(x) &= x + 7 \\ c(m) &= (p_m(0), p_m(1), p_m(2), p_m(3), p_m(4), p_m(5), p_m(6), p_m(7)) \\ c(m) &= (7, 6, 5, 4, 3, 2, 1) \end{aligned}$$

2.6.3 Kod BCH

Kody BCH (Bose-Chaudhuri-Hocquenghem) są podklasą kodów cyklicznych, co oznacza, że każde przesunięcie słowa kodowego jest także słowem kodowym

$$(c_0, c_1, \ldots, c_{n-2}, c_{n-1}),\ (c_{n-1}, c_0, \ldots, c_{n-3}, c_{n-2}), \ldots,\ (c_1, c_2, \ldots, c_{n-1}, c_0)$$

Aby zbudować kod BCH Reeda-Solomona, potrzebujemy najpierw funkcji minimalnej pierwiastka α, czyli takiego minimalnego wielomianu nierozkładalnego $p(x)$ stopnia m, dla którego istnieje element prymitywny α, który pozwala wygenerować całe ciało skończone

$$\mathbb{F}_{2^m} = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{p^m - 1}\}$$

Mając taki element prymitywny, jesteśmy w stanie stworzyć wielomian generujący $g(x)$, używając wzoru

$$t = n - k$$

$$g(x) = \prod_{i=0}^{t-1} (x - \alpha^i) = g_t x^t + g_{t-1} x^{t-1} + \cdots + g_1 x + g_0$$

Aby utworzyć słowo kodowe $c(x)$, należy pomnożyć wielomian wiadomości $p_m(x)$ przez wielomian generujący $g(x)$: $c(x) = p_m(x) \cdot g(x)$

Przykład:

Używając danych z przykładu z sekcji (2.6.1), otrzymujemy:

$$g(x) = \prod_{i=0}^{2}(x - \alpha^i) = (x-1)(x-\alpha)(x-\alpha^2)$$
$$g(x) = x^3 + (\alpha^2 + \alpha + 1)x^2 + (\alpha^2 + 1)x + (\alpha + 1)$$
$$g(x) = x^3 + 7x^2 + 5x + 3$$
$$c(x) = p_m(x) \cdot g(x)$$
$$c(x) = (7x^3 + 6x^2 + 5x + 4) \cdot (x^3 + 7x^2 + 5x + 3)$$
$$c(x) = 7x^6 + 5x^5 + 7x^4 + 3x^3 + 7x^2 + 6x + 7,$$
$$c(x) = (7, 5, 7, 3, 7, 6, 7)$$

2.6.4 Systematyczny kod BCH

Aby uzyskać systematyczne słowo kodowe $c(x)$, musimy obliczyć:

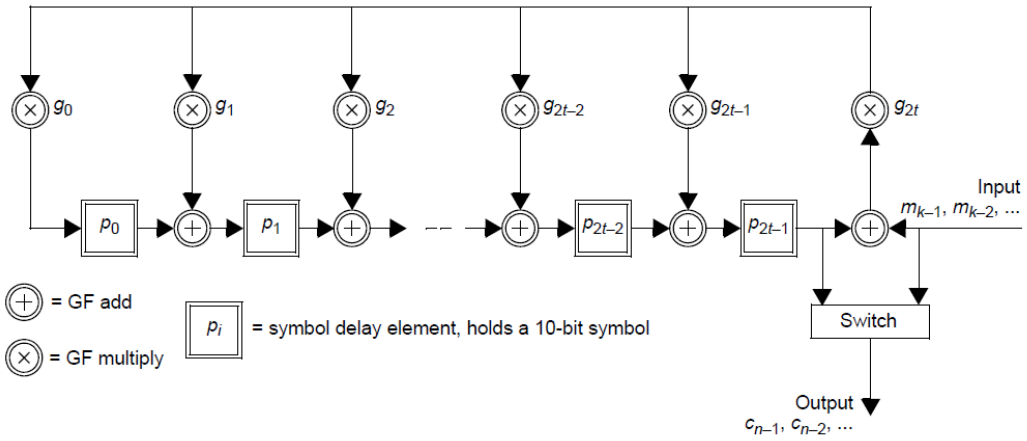
$$c_r(x) = p_m(x) \cdot x^t \mod g(x)$$
$$c(x) = p_m(x) \cdot x^t - c_r(x)$$

Przykład:

Używając danych z przykładu z sekcji (2.6.3), otrzymujemy:

$$c_r(x) = (7x^3 + 6x^2 + 5x + 4) \cdot x^3 \mod x^3 + 7x^2 + 5x + 3$$
$$c_r(x) = 7x^6 + 6x^5 + 5x^4 + 4x^3 \mod x^3 + 7x^2 + 5x + 3$$
$$c_r(x) = 2x + 2$$
$$c(x) = (7x^3 + 6x^2 + 5x + 4) \cdot x^3 - (2x + 2)$$
$$c(x) = 7x^6 + 6x^5 + 5x^4 + 4x^3 + 2x + 2$$
$$c(x) = (7, 6, 5, 4, 0, 2, 2)$$

W standardzie Ethernet znajduje się przykładowa implementacja takiego kodu za pomocą rejestrów przesuwnych, przedstawiona w pracy na rysunku 1, $2t = n - k$



Rysunek 1: Model funkcjonalny kodera RS [3, sekcja 91.5.2.7]

3 Modulacja

3.1 Dlaczego potrzebujemy modulacji?

Modulacja cyfrowa to technika zamiany bitów na sygnał oraz sygnału na bity. Jest kluczowym zagadnieniem w przesyle danych pomiędzy systemami komputerowymi. W odróżnieniu od modulacji analogowej, gdzie przesyłane dane wybierane są z przedziału, modulacja cyfrowa operuje na dyskretnym zbiorze danych (bitach).

Dane reprezentowane są w postaci zmiany parametrów przesyłanego sygnału. Wyróżniane są cztery podstawowe metody:

1. PSK (phase-shift keying) — zmiana fazy fali nośnej sygnału,
2. FSK (frequency-shift keying) — zmiana częstotliwości fali nośnej sygnału,
3. ASK (amplitude-shift keying) — zmiana amplitudy fali nośnej sygnału,
4. QAM (quadrature amplitude modulation) — połączenie PSK oraz ASK, a więc zmieniana jest zarówno amplituda oraz faza.

Zmiany sygnału (symbole) kodujące kolejne bity wybierane są ze skończonego zbioru nazywanego alfabetem modulacji. Dział ten przedstawi popularne techniki modulacji w technologiach Ethernetowych.

3.2 Wprowadzenie

Aby ławiej zrozumieć ideę stojącą za bardziej skomplikowanymi technikami modulacji, należałoby na wprowadzeniu wyjaśnić kilka podstawowych pojęć.

Główną charakterystyką łącza jest szerokość pasma (ang. bandwidth) — określa ona maksymalną (teoretyczną) liczbę bitów jaką łączę jest w stanie przesłać w danym czasie. Podawana jest ona w bitach na sekundę [*bps*] lub w hercach [Hz].

Przepustowość (ang. channel capacity) — rzeczywista szerokość pasma, zmierzona w określonych warunkach.

Przepływność (ang. bit rate) — rzeczywista ilość bitów transmitowanych w jednostce czasu poprzez kanał, podawana również w *bps* lub Hz. Jest stałą charakterystyką danego łącza.

W 1924 roku Harry Nyquist przedstawił światu równanie, za pomocą którego określić można maksymalną przepływność łącza o szerokości pasma B z wykorzystaniem V poziomów:

$$\text{Przepływność}_{max} = 2B * log_2 V$$

24 lata później Claude Shannon rozszerzył równanie Nyquista, uwzględniając szum. Udowodnił on, że maksymalną przepływność łącza o szerokości pasma B oraz stosunku sygnału do szumu S/N można obliczyć ze wzoru:

$$\text{Przepływność}_{max} = B * log_2(1 + S/N)$$

Granica ta nazywana jest limitem Shannona.

Innym ważnym pojęciem jest multipleksacja (ang. multiplexing) i oznacza przesył wielu symboli jednocześnie w jednym kanale — realizowany w postaci kilku przewodów, na które dane podawane są jednocześnie w każdym cyklu zegara.

3.3 Non-Return-to-Zero

Na początku rozważmy przykład. Najprostszą metodą byłoby używanie dodatniego napięcia dla bitu równego 1 i ujemnego napięcia dla 0. Technika ta nosi nazwę **NRZ (Non-Return-to-Zero)**. W praktyce wykorzystywana jest w połączeniu z kodowaniem liniowym np. 64b/66b, ale nie stosuje się tej techniki samoistnie. Jest tak, ponieważ podczas przesyłania danych mogą wystąpić długie ciągi zer lub jedynek, a więc nadawany sygnał nie będzie się zmieniał. Jest to zjawisko niepożądane podczas transmisji i może doprowadzić do desynchronizacji zegarów strony nadawczej i odbiorczej.

Rozważmy też przypadek, w którym nadawane jest naprzemiennie 1 i 0 — otrzymamy wówczas okres równy 2 bity, co oznacza, że potrzebujemy pasma B/2 Hz przy prędkości B bit/s. Nie trudno zauważyć, że do szybszego nadawnia zwiększona musi zostać szerokość pasma, co nie jest optymalnym rozwiązaniem z uwagi na ograniczoność tego zasobu [4].

Jednym z rozwiązań tego problemu jest wykorzystanie większej ilości poziomów napięcia. W powyższym przykładzie zastosowane zostały dwa poziomy, a co za tym idzie mamy do dyspozycji dwa symbole przesyłane przez kanał. Zwiększenie poziomów do 4 dałoby nam 4 różne symbole, a więc 2 bity informacji. W rezultacie przepływność wzrosła dwukrotnie, natomiast szerokość pasma nie zmieniła się. Technika zadziała pod warunkiem, że strona odbiorcza dysponuje sprzętem, który pozwoli jej na rozróżnienie wielu poziomów napięcia. Jednakże w praktyce jest to koszt, który jesteśmy w stanie ponieść.

3.4 Pulse-Amplitude Modulation

Pulse-Amplitude Modulation (PAM) jest jedną z najpopularniejszych technik modulacji wykorzystywaną w technologiach Ethernetowych. Można ją również zobaczyć w innych technologiach (USB4, PCI Express 6.0). Jest to rodzaj modulacji, w którym dane przesyłane jako zmiany amplitudy sygnału. Modulacje PAM można podzielić na dwie kategorię:

1. single polarity PAM — do sygnału dodawana jest stała składowa, aby wartości napięcia były dodatnie

2. double polarity PAM — wartości mogą być ujemne lub dodatnie

Modulacja PAM pozwala na przesył więcej niż jednego bitu w jednym takcie zegara, dzięki czemu zgodnie z równaniem Nyquista, zwiększona może zostać przepływność przy niezmienionej szerokości pasma. Poszczególne techniki PAM różnią się między sobą liczbą wykorzystywanych poziomów modulacji (technika NRZ może zatem być nazwana PAM2). Liczba możliwych poziomów jest nieograniczona, jednak wraz ze wzrostem liczby poziomów, różnica pomiędzy symbolami maleje — a to utrudnia stronie odbiorczej odczyt symboli. Dlatego właśnie rodzaj modulacji PAM wybiera się na podstawie możliwości strony nadawczej i odbiorczej. W systemach wbudowanych czy technologiach automotive wykorzystywanych jest mniej poziomów, ponieważ w tych przypadkach liczy się kompaktowość, a przepływ danych nie jest duży. Zupełnie odwrotnie jest w technologiach Gigabit Ethernet, gdzie nie ma tak restrykcyjnych ograniczeń sprzętowych, a danych do przesyłu jest dużo.

Warto zwrócić uwagę na to, że wykorzystanie większej ilości poziomów nie adresuje problemu długich sekwencji samych zer lub jedynek. Dlatego również tu stosuje się mechaznizmy zamiany wprowadzonych danych na bardziej zróżnicowany ciąg (np. kodowanie liniowe, skrambling).

3.4.1 PAM3

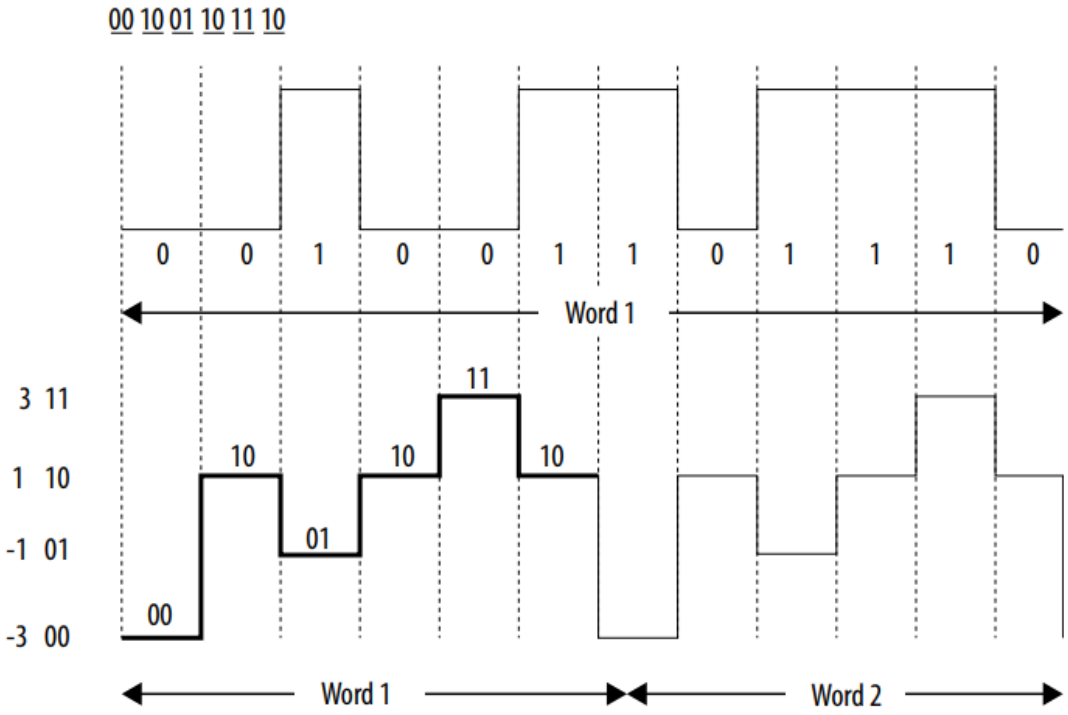
PAM3 — trzypoziomowa modulacja PAM — do przesyłu wykorzystuje wartości $+1$, 0, -1. Jeden symbol koduje $log_2 3 \approx 1,58$ bit/symbol. PAM3 użyty został m.in. w standardach 100BASE-T4 (wczesna implementacja Fast Ethernet) i BroadR-Reach Ethernet standard — wykorzystywany w branży automotive, opracowany przez firmę Broadcom Corporation. Nie jest to schemat modulacji PAM, który często pojawia się w praktycznych rozwiązaniach z uwagi na to, że technologie wykorzystujące tę technikę nie zostały szeroko przyjęte.

Ciekawym aspektem jest podział bitów na symbole. Na pierwszy rzut oka widać, że istnieje problem z grupowaniem bitów — nie można przecież przesłać $1,58$ bita. W modulacjach z liczbą poziomów będącą potęgą 2 podział jest bardzo prosty — bity grupujemy w dwójki, czwórki... i bezpośrednio mapujemy na symbole.

Tutaj rozwiązaniem jest tymczasowa zamiana bitów na trity (system trójkowy). Przykładowo w wersji drugiej USB4 dane dzieli się 11-bitowe grupy, po czym każda z nich zamienana jest na 7 tritów, osiągając przy tym efektywność rzędu $\frac{11/7}{log_2 3} * 100\% \approx 99\%$

3.4.2 PAM4

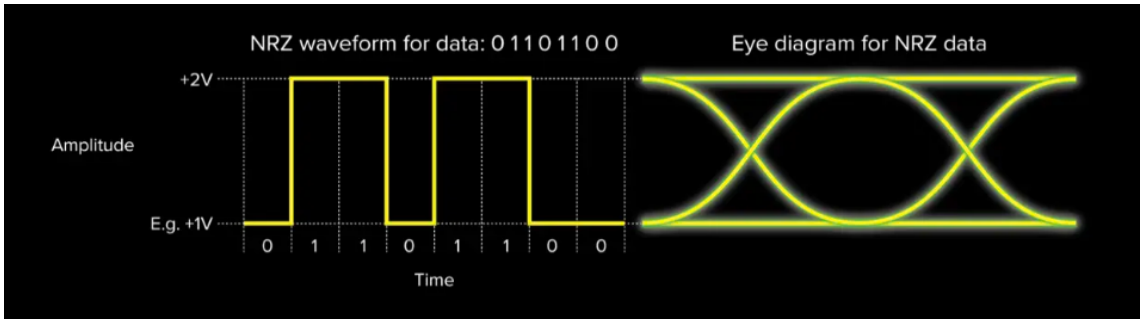
PAM4 — czteropoziomowa modulacja PAM — do przesyłu wykorzystuje wartości 3, 1, -1, -3, które kolejno odpowiadają logicznym wartościom 11, 10, 01, 00. W porównaniu do NRZ (Non-Return-to-Zero) ma przewagę posiadania dwukrotnie większej przepływność przy tej samej prędkości transmisji, co ilustruje poniższy rysunek [5]:



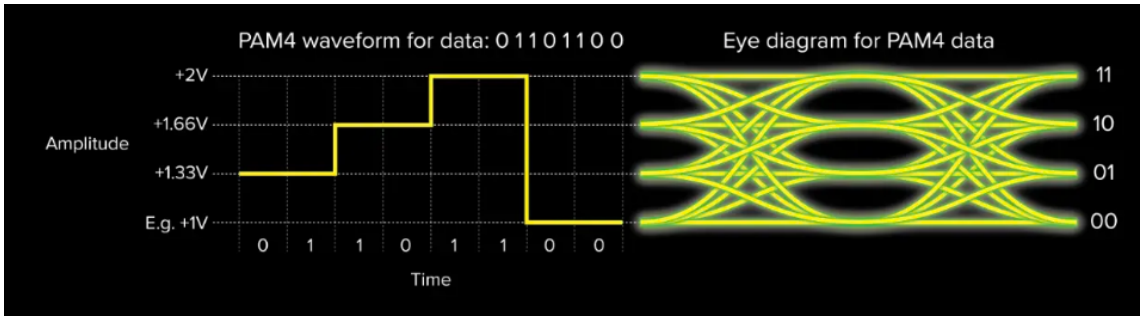
Rysunek 2: Porównanie NRZ oraz PAM4

Warto zwrócić uwagę na to, że w NRZ mamy jedno narastające zbocze ($0 \to 1$) i jedno opadające zbocze ($1 \to 0$), co daje dwie zmiany napięcia. W przypadku PAM4 jest to 6 narastających zbocz ($00 \to 01, 00 \to 10, 00 \to 11, 01 \to 10, 01 \to 11, 10 \to 11$) oraz 6 opadających zbocz ($11 \to 10, 11 \to 01, 11 \to 00, 10 \to 01, 10 \to 00, 01 \to 00$), które łącznie dają 12 różnych zmian napięcia. Ma to znaczący wpływ na stosunek sygnału do szumu (SNR).

PAM4 stał się bardzo popularny w ostatnich latach i jest wykorzystywany m.in. w technologiach 100, 200 i 400 Gigabit Ethernet. Format ten został ustandaryzowany także dla światłowodów (przykładowo 200GBASE-DR4).



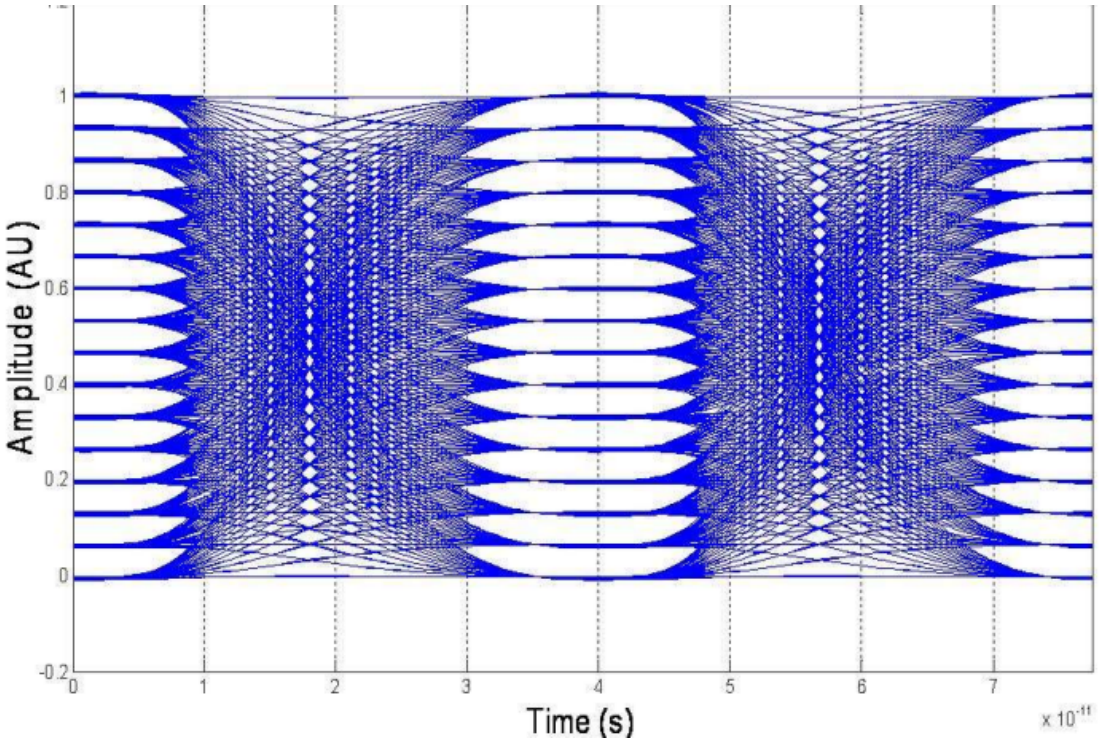
Rysunek 3: Diagram oka NRZ



Rysunek 4: Diagram oka PAM4

3.4.3 PAM16

PAM16 — szesnastopoziomowa modulacja PAM — analogicznie do poprzednich przypadków wykorzystuje wartości 15, 13, 11, ..., 1, −1, −3, ..., −13, −15. Szesnaście poziomów daje 4 bity na symbol. Modulacja PAM16 wykorzystywana jest w technologiach 10GBASE-T, 25GBASE-T czy 40GBASE-T.



Rysunek 5: Diagram oka PAM16

3.4.4 PAM — podsumowanie

Według artykułu [6] przejście z PAM4 na modulacje wyższych rzędów podlega dyskusji. W przypadku PAM4, dwukrotnie większa liczba poziomów niż w NRZ była opłacalna pomimo mniejszego stosunku sygnału do szumu (SNR). Jednakże przejście z PAM4 na PAM16 nie jest pod tym

kątem wydajne, ponieważ opiera się na bardziej wyrafinowanych technikach odczytu i interpretacji przesyłanych symboli, co wymaga większego zużycia energii.

Z drugiej strony modulacje PAM wyższych rzędów pozwalają na zwiększenie przepustowości w systemach z ograniczoną szerokością pasma — co umożliwia implementację technologii wielogigowych w skrętce — dzięki czemu modulacja PAM16 przyjęła się w standardach 10GBASE-T, 25GBASE-T czy 40GBASE-T.

Te same technologie 10/25/40/50 GbE, które wykorzystują światłowód jako medium, a także technologie 200 GbE i 400 GbE używają NRZ oraz PAM4 z uwagi na prostotę.

Tabela 3: Modulacje w różnych standardach Ethernetowych

Standard	Medium	Modulacja
1000BASE-T	skrętka	PAM5
1000BASE-T1	skrętka	PAM3
1000BASE-KX	światłowód	NRZ
10GBASE-T	skrętka	PAM16
10GBASE-SR	światłowód	NRZ
25GBASE-T	skrętka	PAM16
25GBASE-SR	światłowód	NRZ
40GBASE-T	skrętka	PAM16
50GBASE-SR	światłowód	PAM4
100GBASE-ZR	światłowód	PAM4
200GBASE-SR2	światłowód	PAM4
400GBASE-SR4	światłowód	PAM4

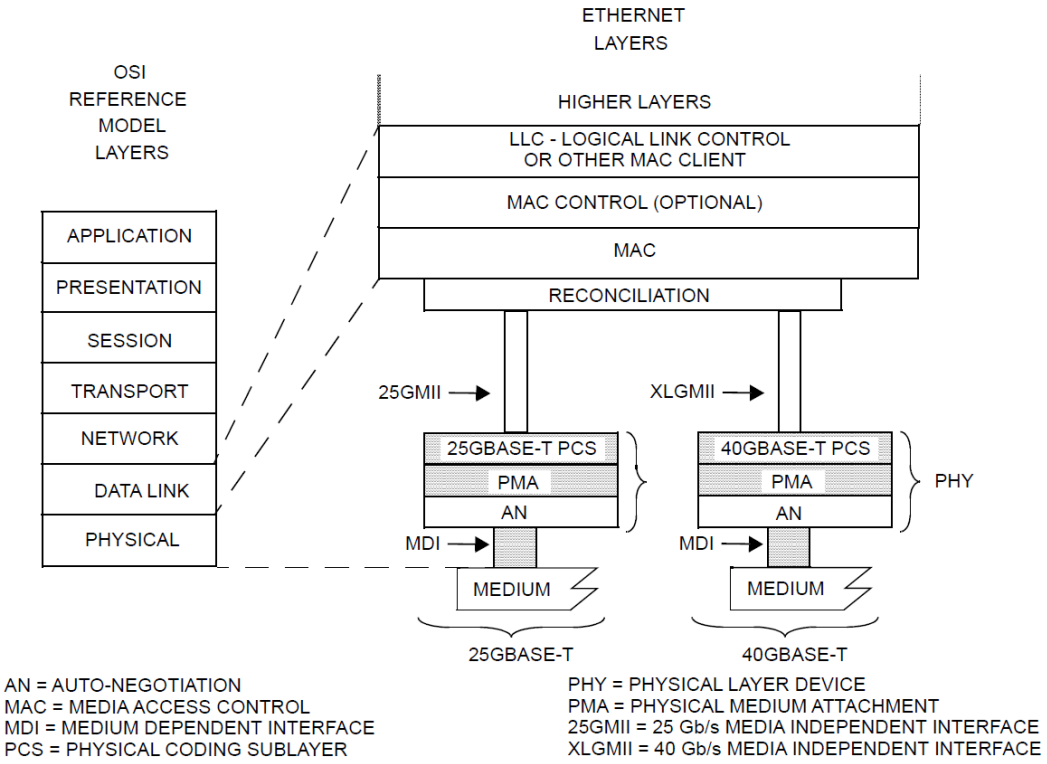
4 40GBASE-T

4.1 Wprowadzenie

40GBASE-T jest technologią transmitowania ramek Ethernetowych z prędkością 40 gigabitów na sekundę, wykorzystując skrętkę jako medium. Technologia została zdefiniowana po raz pierwszy jako część standardu IEEE 802.3ba w 2010 roku.

40GBASE-T zapewnia dwukierunkową komunikację przez cztery przewody skrętki. Każdy przewód transmituje $\frac{1}{4}$ zgromadzonych danych.

4.2 Położenie 40GBASE-T w modelu OSI



Rysunek 6: Położenie 40GBASE-T w modelu OSI

4.3 Media-Independent Interface

MII czyli Media-Independent Interface to interfejs pomiędzy warstwą MAC oraz PHY. Powstał po to, aby warstwy MAC oraz PHY były niezależne od siebie - pozwala na pracę kontrolera MAC z warstwą PHY bez względu na to, jakie medium jest w użyciu. W technologii 40GBASE-T MII określany jest jako XLGMII. Składa się z dwóch wektorów 64-bitowych TXD i RXD - odpowiedzialne za przesył danych w obu kierunkach, dwóch wektorów 8-bitowych TXC i RXC -

wykorzystywane do przesyłu sygnałów kontrolnych oraz TX_CLK i RX_CLK - którymi podawany jest sygnał zegara. XLGMII pozwala na przesył danych na poziomie 40 Gb/s, w obu kierunkach.

4.4 Warstwa PCS

40GBASE-T PCS (Physical Coding Sublayer) jest warstwą odpowiedzialną za kodowanie i dekodowanie oraz skramblowanie i deskramblowanie. W jej skład wchodzą m.in.:

1. skrambler, deskrambler — urządzenie, które modyfikuje (miesza, zamienia w bardziej losowy ciąg) przesyłane dane, aby były lepiej dostosowane do transmisji. Zapewnia synchronizację zegara strony nadawczej i odbiorczej, chroni przed stałą składową, pomaga redukować błędy transmisyjne. Deskrambler zamienia odebrany, zmodyfikowany ciąg w jego oryginalną formę.
2. Koder i dekoder RS-FEC (Reed-Solomon Forward Error Correction).
3. Koder i dekoder LDPC (Low-Density Parity Check) — kod korekcyjny, używany równolegle z kodowaniem Reeda-Solomona.
4. Układ mapujący bity na symbole DSQ128

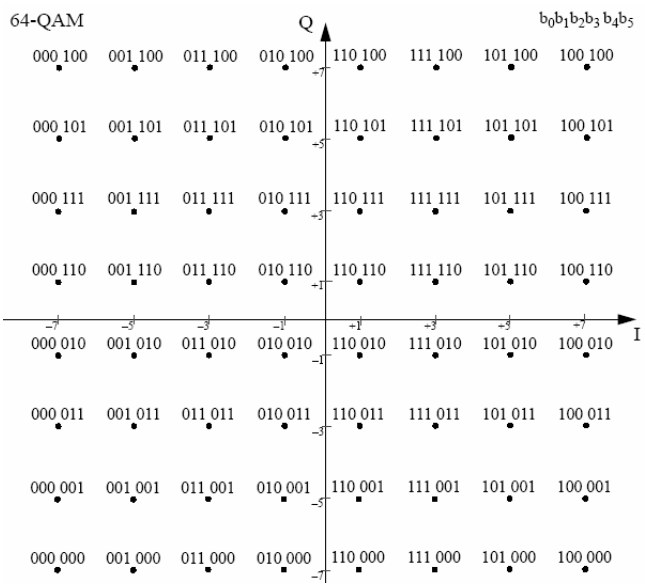
Dane trafiają z warstwy MAC do PCS przez XLGMII i gromadzone są w 64-bitowe bloki. Po zebraniu 50 takich bloków, pierwsze 48 z nich transkodowane są w 512-bitowe bloki, a pozostałe są do nich dołączane. Dane są następnie skramblowane i dołączany jest do nich bit pomocniczy (auxiliary bit), po czym dzielone są na dwa zbiory — pierwszy z nich trafia do kodera Reed-Solomona, a drugi przetwarzany jest przez koder LDPC. Otrzymuje się w ten sposób $512 * 3$ bitów zakodowanych przez RS-FEC oraz $512 * 4$ bitów — LDPC, które łączone są w 7-bitowe grupy (u_0, u_1, u_2), (c_0, c_1, c_2, c_3).

4.5 Modulacja w 40GBASE-T

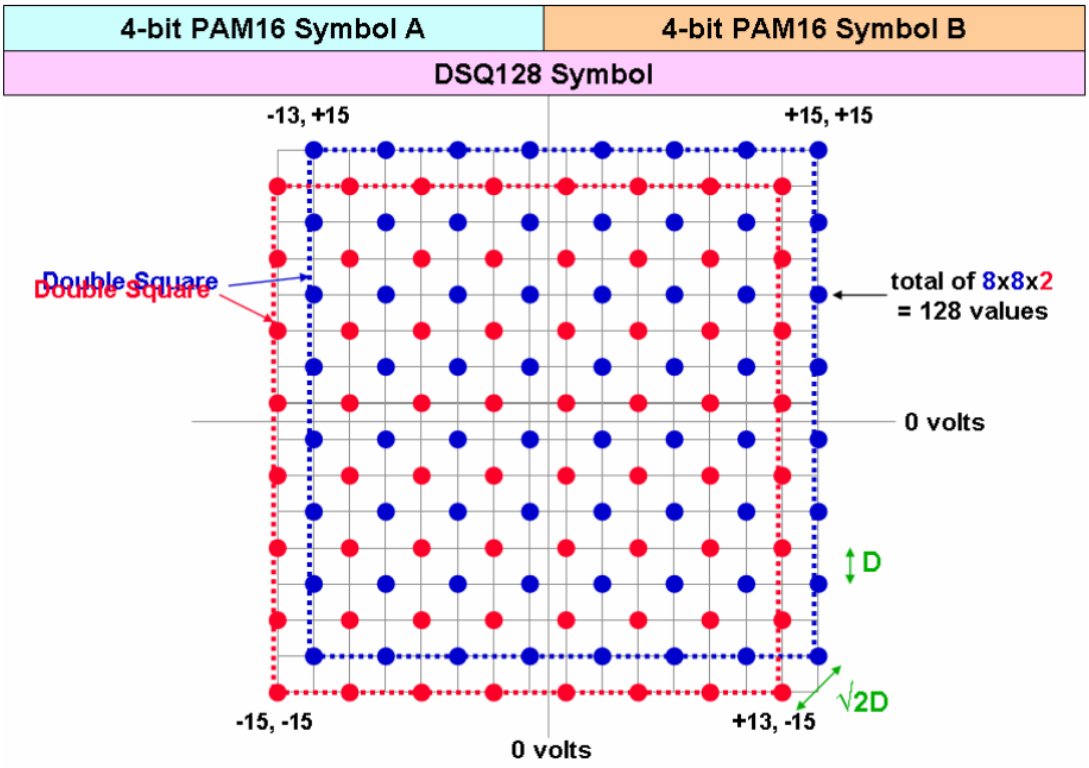
Technologia 40GBASE-T wykorzystuje 16-poziomową modulację PAM - mając 16 różnych poziomów amplitudy możemy zakodować 16 różnych wartości co daje nam 4 bity danych. Oprócz bitów niosących dane, przesyła się również bity pomocnicze (auxiliary bits), przez co w rzeczywistości jeden symbol PAM16 koduje 3,125 bitów informacji. W ciągu każdej sekundy przesyłanych jest 3200 milionów symboli co przekłada się na prędkość transmisji równą 10 Gb/s (3,125 bit/symbol * 3200 MBd) na każdej z czterech par skrętki, co sumarycznie daje transmisję 40 Gb/s.

4.5.1 Double Square Quadrature Amplitude Modulation

Jak opisano w 4.4, warstwa PCS koduje otrzymane dane w 7-bitowe grupy. W technologii 40GBASE-T symbole nadawane przez warstwę PMA wybierane są z konstelacji DSQ128 (Double Square Quadrature Amplitude Modulation).



Rysunek 7: Diagram konstelacji 64QAM



Rysunek 8: Diagram konstelacji DSQ128

Aby wyjaśnić, czym jest DSQ128, pierw spójrzmy na modulację 64QAM. 64QAM (Quadrature Amplitude Modulation) to modulacja, która jest połączeniem modulacji amplitudy oraz fazy. Bity zamieniane są na symbole według diagramu konstelacji rys. 7.

Diagram podzielony jest na 4 sekcje, każda z nich zawiera 16 maksymalnie oddzielonych od siebie punktów. Punkty znajdujące się obok siebie różnią się jednym bitem — dzięki czemu, gdy odbiorca otrzyma zły symbol, najprawdopodobniej tylko jeden bit będzie przekłamany.

DSQ128 jest złożeniem dwóch modulacji 64QAM i składa się z ośmiu sekcji, a każda z nich zawiera 16 punktów (rys.8). Mając 7-bitową grupe (u_0, u_1, u_2), (c_0, c_1, c_2, c_3), pierwsze 3 bity definiują lewy dolny punkt w jednym z 8 regionów, natomiast na podstawie 4 kolejnych bitów wybierany jest jeden z 16 punktów w danym regionie. Proces ten opisany jest szczegółowo w kolejnej sekcji.

4.5.2 Mapowanie ramki LDPC na DSQ128

Zamianę 7-bitów (u_0, u_1, u_2), (c_0, c_1, c_2, c_3) pokazuje poniższy algorytm:

Krok 1:

$$x_{13} = \neg u_0 * u_2$$
$$x_{12} = u_0 \oplus u_2$$
$$x_{11} = c_0$$
$$x_{10} = c_0 \oplus c_1$$
$$x_{23} = (u_1 * u_2) + (u_0 * \neg u_1)$$
$$x_{22} = u_1 \oplus u_2$$
$$x_{21} = c_2$$
$$x_{20} = c_2 \oplus c_3$$

Krok 2:

$$x_1 = 8x_{13} + 4x_{12} + 2x_{11} + x_{10}$$
$$x_2 = 8x_{23} + 4x_{22} + 2x_{21} + x_{20}$$

Krok 3:

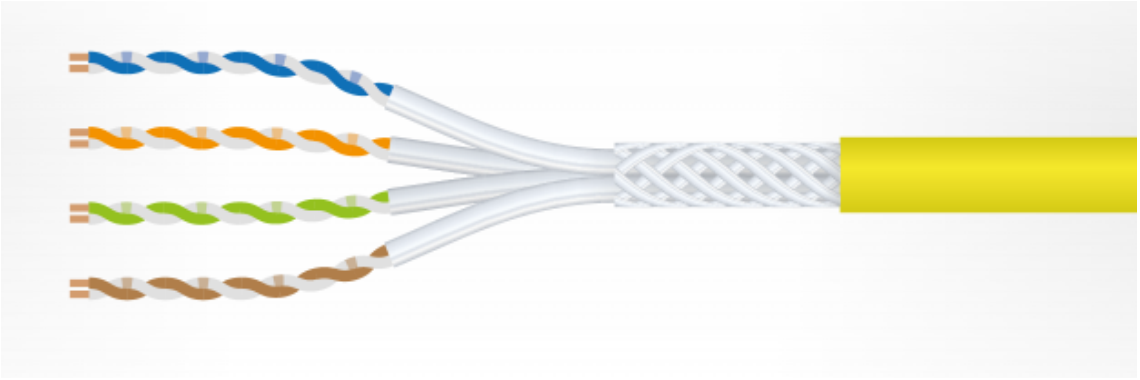
$$y_1 = (x_1 + x_2) \mod 16$$
$$y_2 = (-x_1 + x_2) \mod 16$$

Krok 4:

$$\text{PAM16}_1 = 2y_1 - 15$$
$$\text{PAM16}_2 = 2y_2 - 15$$

Otrzymane w ten sposób symbole są transmitowane na odpowiednich parach skrętki:

Pair A	$PAM16_1<0>$	$PAM16_2<0>$	$PAM16_1<4>$	$PAM16_2<4>$	...	$PAM16_1<508>$	$PAM16_2<508>$
Pair B	$PAM16_1<1>$	$PAM16_2<1>$	$PAM16_1<5>$	$PAM16_2<5>$	...	$PAM16_1<509>$	$PAM16_2<509>$
Pair C	$PAM16_1<2>$	$PAM16_2<2>$	$PAM16_1<6>$	$PAM16_2<6>$	...	$PAM16_1<510>$	$PAM16_2<510>$
Pair D	$PAM16_1<3>$	$PAM16_2<3>$	$PAM16_1<7>$	$PAM16_2<7>$	...	$PAM16_1<511>$	$PAM16_2<511>$



Rysunek 9: Skrętka

5 Symulator

5.1 Język programowania

Do stworzenia symulatora wybrano język programowania Python z uwagi na kilka istotnych powodów. Przede wszystkim, czytelność składni stanowi ogromne ułatwienie podczas wspólnego tworzenia oprogramowania, a prostota pozwala skupić się na istocie problemu, nie tracąc czasu na pokonywanie trudności języka.

Dodatkowo wybór Pythona jest motywowany chęcią rozwijania naszych umiejętności w tym środowisku zarówno na poziomie indywidualnym, jak i zawodowym. Python cieszy się dużą popularnością jako uniwersalny język programowania, używany w różnych dziedzinach, takich jak analiza danych, sztuczna inteligencja czy aplikacje webowe. Posiadanie umiejętności programowania w Pythonie otwiera drzwi do szerszych możliwości zawodowych i dostępu do różnorodnych ciekawych projektów.

Jednym z najważniejszych argumentów przemawiających za wyborem Pythona jest jego ogromna popularność. Związana z tym społeczność programistyczna tworzy rozbudowany ekosystem, oferujący dostęp do wielu gotowych rozwiązań, bibliotek i frameworków. W kontekście tworzenia symulatora istnieje wiele bibliotek w Pythonie, które mogą okazać się niezwykle przydatne. Na przykład biblioteki umożliwiające tworzenie interfejsów graficznych ułatwią korzystanie z symulatora, biblioteki do analizy i przetwarzania sygnałów pomogą modelować różne aspekty transmisji, a biblioteki do wizualizacji pozwolą na przedstawienie wyników w przystępny sposób.

Inną cechą, która wyróżnia ten język programowania, jest jego przenośność. To ma dla nas duże znaczenie przy tworzeniu symulatora, który musi działać w warunkach laboratoryjnych, a więc na dowolnym popularniejszym systemie operacyjnym oraz charakteryzować się łatwością instalacji. Te wymagania Python w naszej ocenie spełnia.

5.2 Narzędzia, biblioteki i moduły

Jednym z celów postawionych przez promotora jest wykorzystanie gotowych rozwiązań podczas pracy nad symulatorem. W tym rozdziale zostaną przedstawione biblioteki i moduły języka Python oraz inne narzędzia, które mogą zostać wykorzystane w programie.

Python oferuje wiele bibliotek, które mogą okazać się kluczowe: od interfejsu graficznego po gotowe narzędzia do symulacji. Oto przegląd kilku z nich, które brano pod uwagę przy projektowaniu rozwiązania:

1. PyQt będzie biblioteką wykorzystywaną do stworzenia interfejsu graficznego użytkownika (GUI) dla symulatora. PyQt zapewnia szeroki zakres narzędzi do tworzenia rozbudowanych i przyjaznych użytkownikowi interfejsów, co jest szczególnie ważne w symulatorze dydaktycznym, gdzie interfejs musi być intuicyjny i nie stanowić niepotrzebnego wyzwania lub

problemu dla biorących udział studentów

2. NumPy jest najpopularniejszą biblioteką Python implementującą algorytmy matematyczne. Między innymi oferuje generatory liczb pseudolosowych o różnych rozkładach, co jest wymagane do prawidłowego generowania ramek ethernetowych i błędów

3. Matplotlib to popularna biblioteka do tworzenia wykresów. Może okazać się przydatna przy tworzeniu wykresów sygnałów

Istnieją również inne popularne narzędzia, które mogą być użyteczne do symulacji rozwiązań warstwy fizycznej sieci Ethernet:

1. SPICE (Simulation Program with Integrated Circuit Emphasis) jest powszechnie stosowanym narzędziem do symulacji obwodów elektronicznych. Jest to rozbudowany program, który umożliwia modelowanie i analizę zachowania obwodów złożonych, takich jak układy analogowe, cyfrowe czy mikroelektroniczne. W celu korzystania z tego narzędzia w środowisku Python dostępna jest biblioteka PySpice, będąca interfejsem umożliwiającym korzystanie ze SPICE,

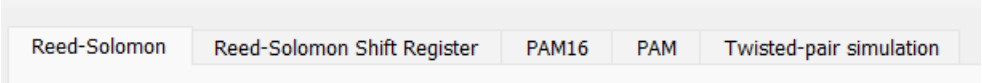
2. MATLAB to znane i powszechnie używane narzędzie do obliczeń numerycznych, analizy danych i modelowania systemów. Posiada szeroki zakres narzędzi i funkcji przeznaczonych do tworzenia modeli matematycznych, symulacji dynamicznych itp.,

3. Scapy umożliwia tworzenie i przetwarzanie różnego rodzaju pakietów sieciowych, w tym ramek Ethernet, co jest kluczową funkcjonalnością symulatora.

5.3 Interfejs użytkownika

Interfejs użytkownika został wykonany przy użyciu PyQt5 oraz Qt Designer. Qt Designer to graficzne narzędzie do projektowania interfejsów użytkownika w ramach frameworka Qt. Umożliwia łatwe tworzenie i dostosowywanie wyglądu aplikacji oraz następne jego wygenerowanie jako kodu w języku Python lub C++.

Interfejs składa się z kilku zakładek (Rysunek 10), które umożliwiają przełączanie między częściami aplikacji bez utraty wyników dotychczasowej pracy. Każda zakładka przeznaczona jest do innego zadania laboratoryjnego i zawiera symulacje innych rozwiązań ethernetowych.



Rysunek 10: Zakładki symulatora

Wykresy przedstawione na Rysunku 15 są tworzone przy pomocy biblioteki Matplotlib. Została dodatkowo stworzona klasa, która zawiera stworzone wykresy i może być użyta jako element graficznego interfejsu użytkownika, a więc dodana do niego.

5.4 Symulacje wybranych rozwiązań

Aplikacja umożliwa symulowanie wybranych rozwiązań fizycznej warstwy sieci Ethernet. W każdym przypadku użytkownik ma swobodę podawania własnych parametrów wejściowych, zmieniania ich, co ma na celu lepsze zrozumienie działania tych rozwiązań.

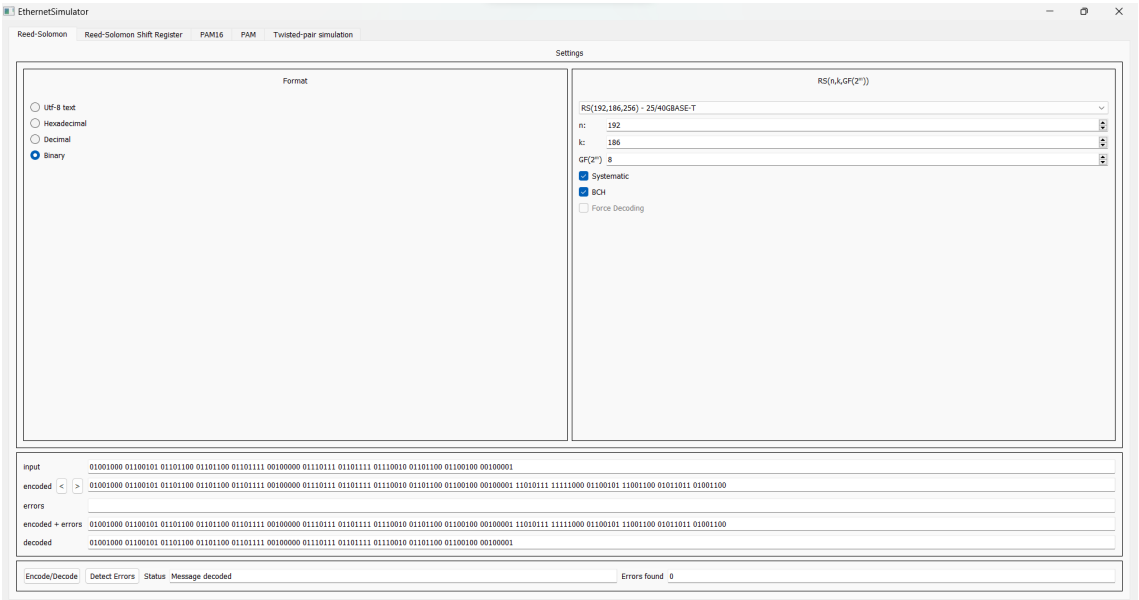
5.4.1 Kodowanie korekcyjne Reeda-Solomona

Kodowanie korekcyjne Reed-Solomona to metoda kodowania korekcyjnego, mająca na celu wykrywanie i naprawianie błędów w przesyłanych danych. Stworzona została w 1960 roku przez dwóch amerykańskich matematyków: Irving S. Reed i Gustave Solomon. Od tego czasu znalazła szerokie zastosowanie w dziedzinie komunikacji, kodowaniu i obsłudze dysków.

Główną ideą kodowania korekcyjnego Reed-Solomona jest dodawanie nadmiarowych danych do przesyłanych informacji, dzięki czemu w przypadku wystąpienia błędów, możliwe jest ich wykrycie i skorygowanie. Algorytm opiera się na algebraicznych właściwościach ciał skończonych, co umożliwia efektywne wykonywanie operacji matematycznych potrzebnych do kodowania i dekodowania.

W symulatorze kodowanie i dekodowanie wykorzytuje metody klasy ReedSolomon udostępnionej w bibliotece galois. Jest ona rozszerzeniem, dodającym operacje na ciałach skończonych, innej popularnej biblioteki języka Python - NumPy. Jej nazwa pochodzi od nazwiska francuskiego matematyka Évariste Galois, który zasłynął badaniami ciał skończonych, które nazywane są również ciałami Galois.

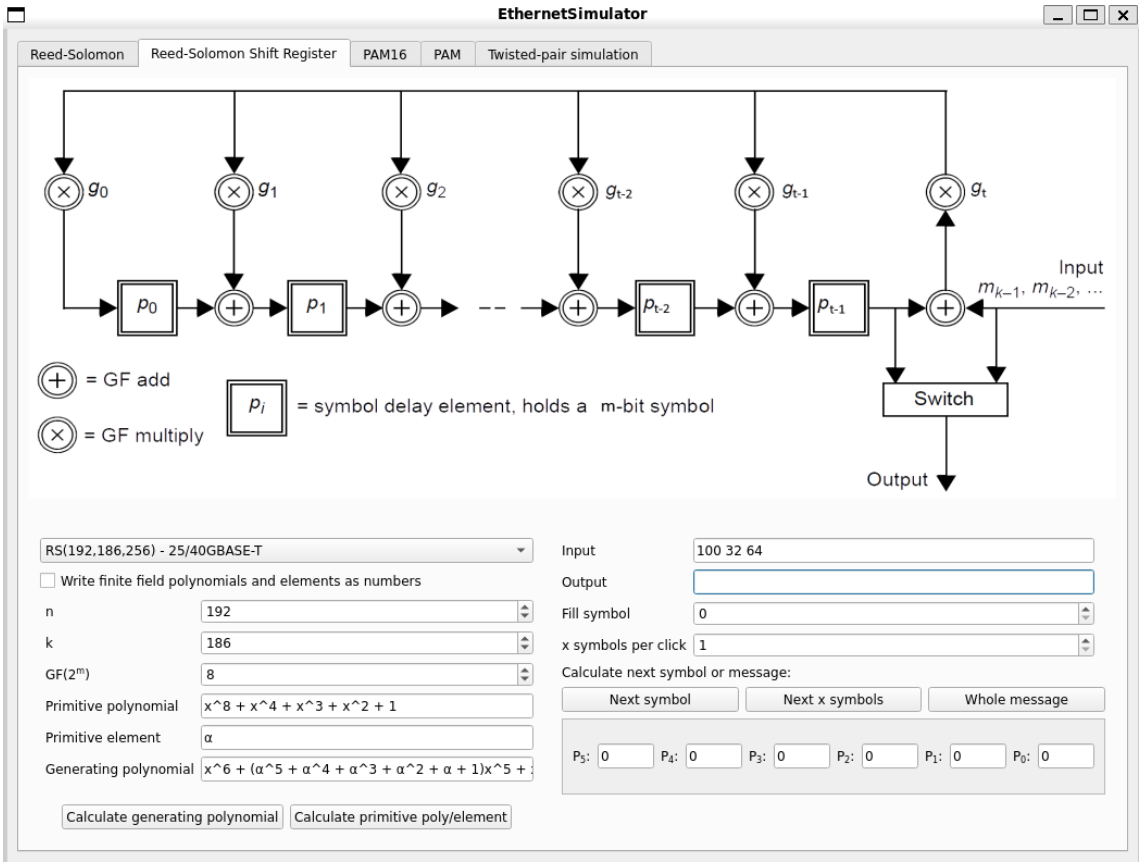
Zakładkę z kodowaniem Reeda-Solomona przedstawia Rysunek 11.



Rysunek 11: Zakładka z kodowaniem Reeda-Solomona

5.4.2 Rejestr przesuwający dla kodowania Reeda-Solomona

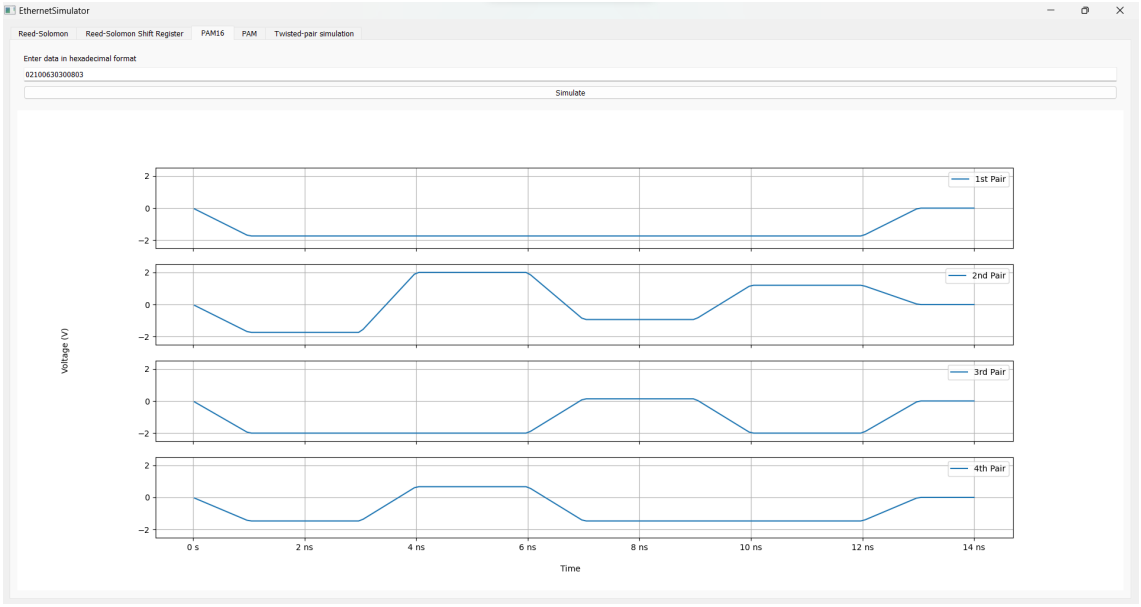
W tej zakładce (Rysunek 12) koder RS został zaimplementowany zgodnie z modelem funkcyjnym udostępnionym w standarcie Ethernet. Po przejściu wszystkich symboli wiadomości element 'Switch' zaczyna przepuszczać symbole parzystości. W opcjach po lewej stronie możemy podobnie jak w poprzedniej zakładce wybrać parametry kodera oraz dodatkowo wybrać inne wielomiany i elementy prymitywne. Przycisk 'Calculate generating polynomial' oblicza wielomian generujący, a 'Calculate primitive poly/ement' oblicza element i wielomian prymitywny dla podanego ciała skończonego $\mathbb{F}_{2^m}$. Po prawej stronie mamy dane wejściowe oraz aktualny stan rejestrów p_i. 'Fill symbol' jest symbolem, który będzie wysyłany, jeżeli zabraknie symboli na wejściu.



Rysunek 12: Zakładka z rejestrem przesuwającym kodowania Reeda-Solomona

5.4.3 PAM16

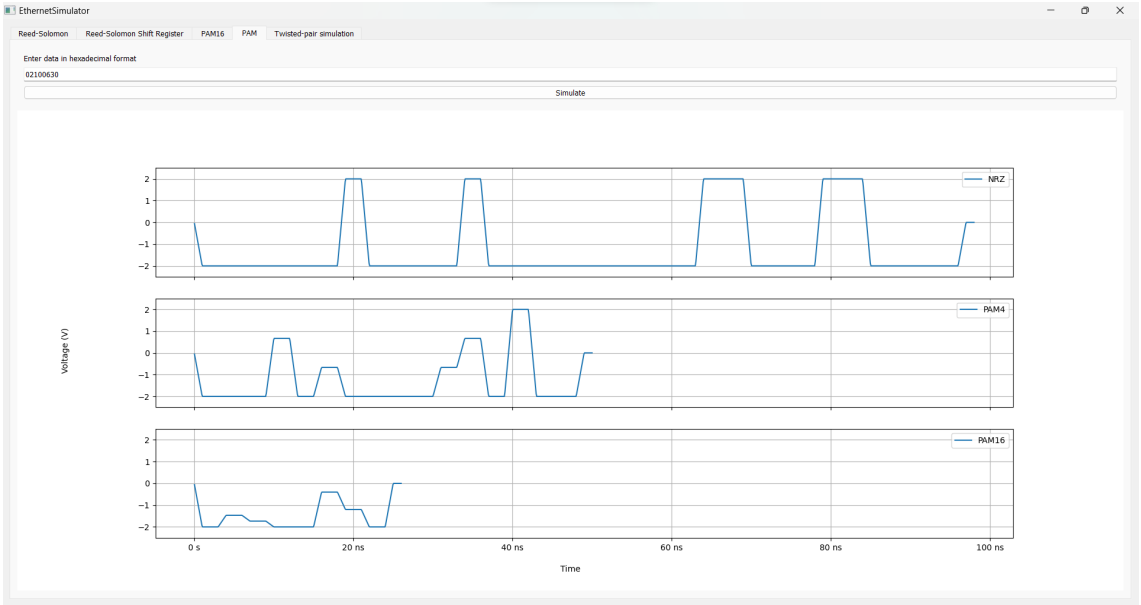
Zakładka PAM16 (Rysunek 13) przedstawia przesył danych podanych w formacie szesnastkowym przez użytkownika przez czteroparową skrętkę.



Rysunek 13: Zakładka z PAM16

5.4.4 PAM

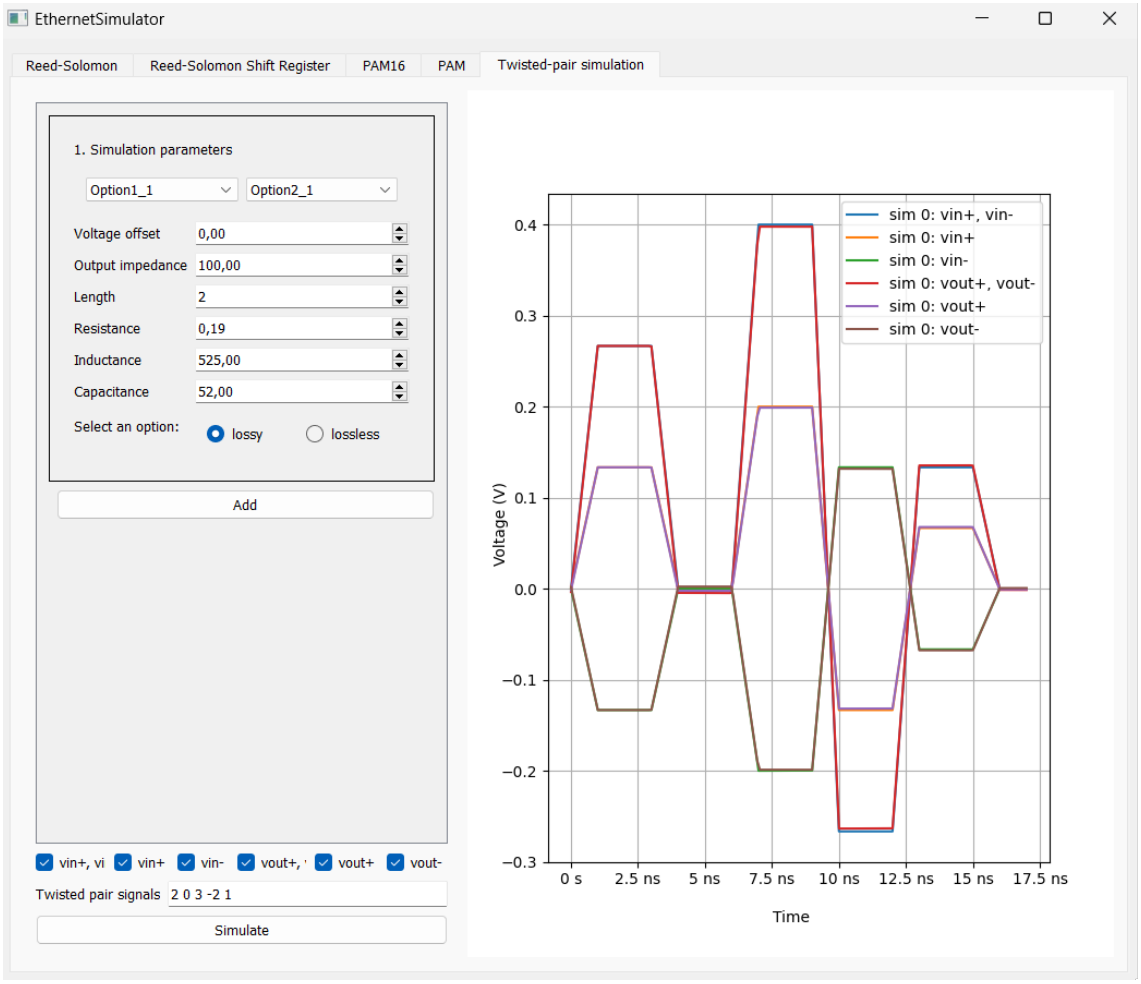
Zakładka PAM (Rysunek 14) przedstawia różnice modulacji: NRZ, PAM4, PAM16. Symulacje są tworzone tak, jak w poprzedniej zakładce, na podstawie danych w formacie szesnastkowym podanych przez użytkownika.



Rysunek 14: Zakładka z różnymi modulacjami

5.4.5 Symulacja przesyłu sygnału

Zakładka Twisted-pair simulation (Rysunek 15) pozwala na śledzenie zmian napięć podczas przesyłu danych przez pojedynczą skrętkę. W tym przypadku wykorzystana została biblioteka PySpice. Dzięki niej można nadać wykorzystywanemu przewodowi pożądane parametry, między innymi: opór, długość i indukcyjność.



Rysunek 15: Zakładka z symulacją

Użytkownik ma możliwość podawania tych parametrów w przewijalnym oknie, gdzie może dodawać również kolejne przewodniki. Patrz Tabela 4.

Tabela 4: Parametry symulacji

Parametr	**Od**	**Do**	**Domyślnie**
Voltage offset[1] [V]	<0	10>	0
Output impedance[2] [Ω]	(0	200>	100
Length[3] [m]	<1	100>	2
Resistence[4] [Ω]	(0	100>	0.19
Inductance[5] [nH]	(0	1000>	525
Capacitance[6] [pF]	(0	100>	52

Symulacja wykonywana jest w osobnym wątku. Dodatkowo możliwa jest jednoczesna symulacja wielu przewodów o różnych parametrach, które następnie przedstawiane są na jednym wykresie. Opcjonalnie można ukrywać lub pokazywać wybrane symulacji zaznaczając odpowiednie pola przy listach parametrów odpowiednich przewodów.

5.4.6 Uruchamianie

W celu uruchomienia poprawnie zainstalowanego symulatora wystarczy wpisać w terminalu: phyether. Dodatkowo możliwe jest podanie dwóch parametrów określających wielkość okna: phyether <szerokość> <wysokość>.

5.4.7 Dokumentacja oprogramowania

Uznano za wystarczającą dokumentację strukturę projektu, wykorzystane nazwy i wyczerpujące komentarze. Projekt ma prostą strukturę oraz jest łatwo rozszerzalny, ponieważ każda symulacja stanowi odrębny moduł. Dodatkowe informacje dotyczące prawidłowej pracy z kodem programu zawarte są w pliku README.md dołączonym do kodu źródłowego.

[1]Napięcie niezrównoważnienia - napięcie, które musi być przyłożone do wejścia, aby wyjście wynosiło 0.

[2]Impendacja wyjściowa - stosunek napięcia wyjściowego do natężenia prądu wyjściowego układu.

[3]Długość skrętki.

[4]Rezystancja - wielkość charakteryzująca relację między napięciem a natężeniem prądu elektrycznego w obwodach.

[5]Indukcyjność - zdolność obwodu do wytwarzania strumienia pola magnetycznego powstającego w wyniku przepływu przez obwód prądu elektrycznego.

[6]Pojemność elektryczna – wielkość fizyczna opisująca zdolność ciała do gromadzenia ładunku elektrycznego.

6 Zajęcia dydaktyczne — sprawozdanie

6.1 Wprowadzenie

Jak wspomniano wcześniej, zakres pracy inżynierskiej obejmuje przeprowadzenie zajęć laboratoryjnych opisanych w dodatku A. Pierwsze zajęcia miały charakter testowy — poza dydaktyką chciano: wychwycić potencjalne błędy w opracowanym symulatorze; sprawdzić czy studenci potrafią zrealizować zadania laboratoryjne po przeczytaniu instrukcji; przygotować dyplomantów na problemy, które mogą pojawić się podczas prowadzenia zajęć (awaria sprzętu, niekompatybilność oprogramowania na różnych systemach operacyjnych, niemożność zainstalowania oprogramowania ze względu na brakujące biblioteki lub wersje bibiotek na komputerach). Problemy, które pojawiły się w trakcie pierwszych zajęć, zostały wychwycone przez autorów, ażeby ostateczna wersja była możliwie najlepsza.

6.2 Przygotowanie zajęć

6.2.1 Stan początkowy

Przygotowania sali do zajęć dydaktycznych z grupą studentów rozpoczęły się pięć dni przed planowanymi zajęciami. Sala jest laboratorium dyplomowym dostępnym dla dyplomantów wydziału ETI Politechniki Gdańskiej. W momencie, gdy rozpoczęto przygotowania dostępnych było osiem komputerów stacjonarnych. Różniły się one zainstalowanymi systemami operacyjnymi: na połowie był to Windows 10, a na połowie Ubuntu 18.

6.2.2 Podjęte działania

W pierwszej kolejności spróbowano zainstalować i uruchomić symulator na komputerze z Ubuntu 18. Instalacja przez Internet okazała się niemożliwa ze względu na nieprzygotowanie zależności pod specyfikę tego systemu operacyjnego. Windows 10 okazał się bezproblemowy i, po użyciu dwóch poleceń: instalacja i uruchomienie symulatora, symulator był gotowy do używania.

W związku z brakiem pakietu libngspice0 na Ubuntu 18 który jest potrzebny do uruchomienia symulacji skrętki zdecydowano się zainstalować na tych komputerach Linux Mint. Po tym czasochłonnym procesie symulator uruchamiał się, jednak napotkano kolejny problem - okno symulatora wykraczało poza ekran i nie było możliwości jego zmniejszenia. Winą była wykorzystana w programie ilustracja. Naprawiono to dodając do polecenia uruchomienia symulatora dwa opcjonalne parametry określające maksymalny rozmiar tej ilustracji.

W tym momencie sala była w ocenie prowadzących gotowa do przeprowadzenia zajęć. Ostatecznie dostępne były cztery komputery z Windows 10 oraz cztery komputery z Linux Mint.

6.2.3 Problemy

Podczas przygotowywania sali napotkano następujące problemy:

- W sali istnieje problem z dostępem do Internetu na więcej niż jednym komputerze,
- Monitory dostępne w sali miały specyficzne proporcje,
- Dla Ubuntu 18 nie było w repozytorium pakietu libngspice0
- Na Linux Mint okno programu rozszerzało się do wymiarów większych niż dostępna wielkość monitora bez możliwości zmniejszenia,
- Aplikacja nie instalowała się na Pythonie 3.12 z powodu niekompatybilności modułów od których zależy symulator

6.3 Przebieg zajęć

Przed zajęciami wprowadzono ostatnie zmiany i poprawki do wejściówki oraz zadań laboratoryjnych. Wydrukowano je w nadmiarowej liczbie, by promotor, recenzent i prowadzący również mieli do nich dostęp.

Laboratoria odbyły się na wydziale ETI Politechniki Gdańskiej. Na początku zrealizowano wprowadzenie teoretyczne do omawianych zagadnień w postaci godzinnego wykładu, który został przeprowadzony przez dyplomantów dla grupy studentów pod okiem promotora.

Następnie podzielono studentów na dwie grupy: pierwsza odbyła zajęcia tego samego dnia, druga — tydzień później. Świadomą decyzją prowadzących uczestnicy zajęć nie byli uprzedzeni o systemach operacyjnych dostępnych na poszczególnych komputerach, więc zajmowali miejsca losowo. Zajęcia rozpoczęto od wejściówki (Dodatek B), a następnie zrealizowano zadania opracowane w ramach pracy (Dodatek C).

W trakcie zajęć studenci wykonywali zadania samodzielnie, konsultując się przy tym między sobą. Autorzy niniejszej pracy byli do ich dyspozycji, po to aby rozwiać wszelkie nieścisłości, nakierować studentów na właściwe rozwiązanie i tym podobne.

Uczestnicy zajęć mieli okazję zaznajomić się z zagadnieniami, które występują w warstwie fizycznej Ethernet, natomiast opracowany symulator pomógł w wizualizacji tychże rozwiązań — porównanie zachowania sygnału w przypadku różnych metod modulacji oraz sprawdzenie działania i właściwości kodera Reeda-Solomona. Rysunek 16 przedstawia zdjęcie wykonane podczas pierwszych zajęć.

Rysunek 16: Zajęcia laboratoryjne przeprowadzone 6 grudnia 2023 r.

Za całość laboratorium studenci mogli uzyskać maksymalnie 4 punkty, w tym: 2 punkty za wejściówkę oraz 2 punkty za zadania laboratoryjne. Prace zostały ocenione w ciągu dwóch dni.

6.3.1 Problemy

Podczas zajęć również napotkano pewne problemy:

- Przed zajęciami należało wgrać najnowszą wersję symulatora, jednak przez problemy z dostępem do Internetu okazało się to tak czasochłonne, że zajęcia rozpoczęły się z opóźnieniem około 15 minut,
- Studenci znaleźli błąd w symulacji PAM, który został naprawiony przed zajęciami z kolejną grupą.

6.3.2 Wnioski

Zarówno instrukcja, jak i symulator okazały się zrozumiałe dla studentów mających z nimi styczność pierwszy raz. System operacyjny nie miał żadnego wpływu na pracę studentów, ponieważ całość odbywała się w oknie symulatora. Zadania były wykonywane sprawnie, a poza pytaniami uczestnicy byli w stanie prowadzić dyskusję z prowadzącymi na temat rozwiązań i treści zadań.

Studenci otrzymali bardzo dobre oceny za wykonaną pracę, uśredniając: 95.9% za całość, 97.2% za wejściówkę oraz 94.6% za część laboratoryjną. Spośród zadań na wejściówce jedyne problemy sprawiały zadania 5 i 6, a z zadań laboratoryjnych nieznacznie trudniejsze okazały się zadania związane z kodowaniem Reeda-Solomona.

7 Podsumowanie

Wszystkie przyjęte cele pracy zostały osiągnięte, a więc:

- stworzono symulator, który umożliwia następujące symulacje:
  - symulację sygnału w skrętce z możliwością modyfikacji parametrów wejściowych oraz kanału,
  - symulację modulacji NRZ, PAM4 oraz PAM16,
  - symulację kodowania korekcyjnego Reeda-Solomona,
- przygotowano instrukcję, wejściówkę oraz zadania laboratoryjne,
- przeprowadzono zajęcia dydaktyczne, wykorzystując podane powyżej materiały.

Dodatkowym sukcesem można określić intuicyjność symulatora, która była dla autorów istotnym założeniem podczas pracy nad rozwiązaniem. Uczestnicy zajęć nie zgłaszali żadnych problemów w obsłudze programu.

Przy planowaniu celów pracy zrezygnowano z pewnych elementów, jednak warto byłoby rozwinąć dyplom o:

- symulację skramblowania i deskramblowania,
- symulację kodowania LDPC,
- symulację przesyłu sygnału światłowodem.

Literatura

[1] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[2] León van de Pavert. Reed-solomon encoding and decoding. Bachelor's thesis, Turku University of Applied Sciences, 2011.

[3] Ieee standard for ethernet. *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pages 1–7025, 2022.

[4] Andrew Tanenbaum Nick Feamster David Wetherall. *Computer Networks, Sixth edition.* Pearson, Mar. 3, 2021.

[5] Intel Corp. An 835: Pam4 signaling fundamentals, Dec. 3, 2019.

[6] Xi Chen Di Che. Higher-order modulation vs faster-than-nyquist pam-4 for datacenter im-dd optics: An air comparison under practical bandwidth limits. *Journal of Lightwave Technology*, 40(10):1–11, 2022.

Dodatek A: Instrukcja

Wejściówka

Na samym początku zajęć odbędzie się wejściówka składająca się z pytań otwartych i zamkniętych. Przykładowe pytania, które mogą znaleźć się na wejściówce to:

1. Podaj definicję dodawania i mnożenia w $\mathbb{F}_2$ bądź wypisz wynik tych działań dla wszystkich możliwych kombinacji elementów.
2. Czym się różni słowo kodowe wygenerowane kodem systematycznym i niesystematycznym
3. Ile błędnych symboli jest w stanie wykryć lub poprawić kod Reeda-Solomona

   A: wykryć: $n-k$, poprawić: $n-k-1$

   B: wykryć: $n-k-1$, poprawić: $n-k-1$

   C: wykryć: $\lfloor \frac{n-k}{2} \rfloor$, poprawić: $\lfloor \frac{n-k}{2} \rfloor$

   D: wykryć: $n-k$, poprawić: $\lfloor \frac{n-k}{2} \rfloor$
4. Podaj zaletę oraz wadę stosowania większej ilości poziomów w modulacjach PAM
5. Czym się różni szerokość pasma od przepustowości?
6. Opisz krótko czym jest NRZ (Non-Return-to-Zero)
7. Czym jest przepływność łącza?

Narzędzia

Symulator - wybór technologii

Do stworzenia symulatora wybrano język programowania Python i wykorzystano między innymi następujące gotowe rozwiązania:

1. PyQt będzie biblioteką wykorzystywaną do stworzenia interfejsu graficznego użytkownika (GUI) dla symulatora. PyQt zapewnia szeroki zakres narzędzi do tworzenia rozbudowanych i przyjaznych użytkownikowi interfejsów, co jest szczególnie ważne w symulatorze dydaktycznym, gdzie interfejs musi być intuicyjny i nie stanowić niepotrzebnego wyzwania lub problemu dla biorących udział w zajęciach,
2. NumPy jest najpopularniejszą biblioteką Python implementującą algorytmy matematyczne. Między innymi oferuje generatory liczb pseudolosowych o różnych rozkładach, co jest wymagane do prawidłowego generowania ramek ethernetowych i błędów,

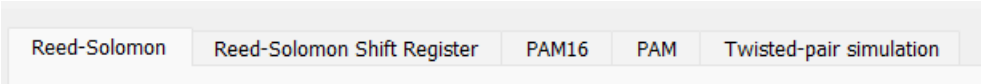
3. Matplotlib to popularna biblioteka do tworzenia wykresów, przydatna przy tworzeniu wykresów sygnałów,

4. SPICE (Simulation Program with Integrated Circuit Emphasis) jest powszechnie stosowanym narzędziem do symulacji obwodów elektronicznych. Jest to rozbudowany program, który umożliwia modelowanie i analizę zachowania obwodów złożonych, takich jak układy analogowe, cyfrowe czy mikroelektroniczne. W celu korzystania z tego narzędzia w środowisku Python dostępna jest biblioteka PySpice, będąca interfejsem umożliwiającym korzystanie ze SPICE,

Interfejs użytkownika

Interfejs użytkownika został wykonany przy użyciu PyQt5 oraz Qt Designer. Qt Designer to graficzne narzędzie do projektowania interfejsów użytkownika w ramach frameworka Qt. Umożliwia łatwe tworzenie i dostosowywanie wyglądu aplikacji oraz następne jego wygenerowanie jako kodu w języku Python lub C++.

Interfejs składa się z kilku zakładek stworzonych, które umożliwiają przełączanie między częściami aplikacji bez utraty wyników dotychczasowej pracy. Każda zakładka przeznaczona jest do innego zadania laboratoryjnego i zawiera symulacje innych rozwiązań ethernetowych.



Rysunek 17: Zakładki symulatora

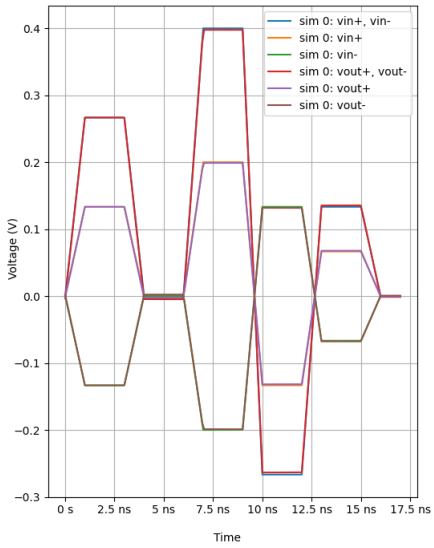
Wykresy są tworzone przy pomocy biblioteki Matplotlib. Została dodatkowo stworzona klasa, która zawiera stworzone wykresy i może być użyta jako element graficznego interfejsu użytkownika, a więc dodana do niego.

Ćwiczenie dydaktyczne — modulacje PAM

W trakcie zajęć laboratoryjnych student będzie miał do dyspozycji symulator phyether, który zawiera implementację wybranych rozwiązań części standardów 25GBASE-T i 40GBASE-T.

Wstęp teoretyczny

Modulacja cyfrowa to technika zamiany bitów na sygnał oraz sygnału na bity. Jest kluczowym zagadnieniem w przesyle danych pomiędzy systemami komputerowymi. Technologie Ethernetowe wykorzystują wiele technik modulacji. Ćwiczenie te ma na celu przybliżenie oraz porównanie występujących technik modulacji PAM (Pulse-Amplitude Modulation), która jest jedną z najpopularniejszych w technologii Ethernet.



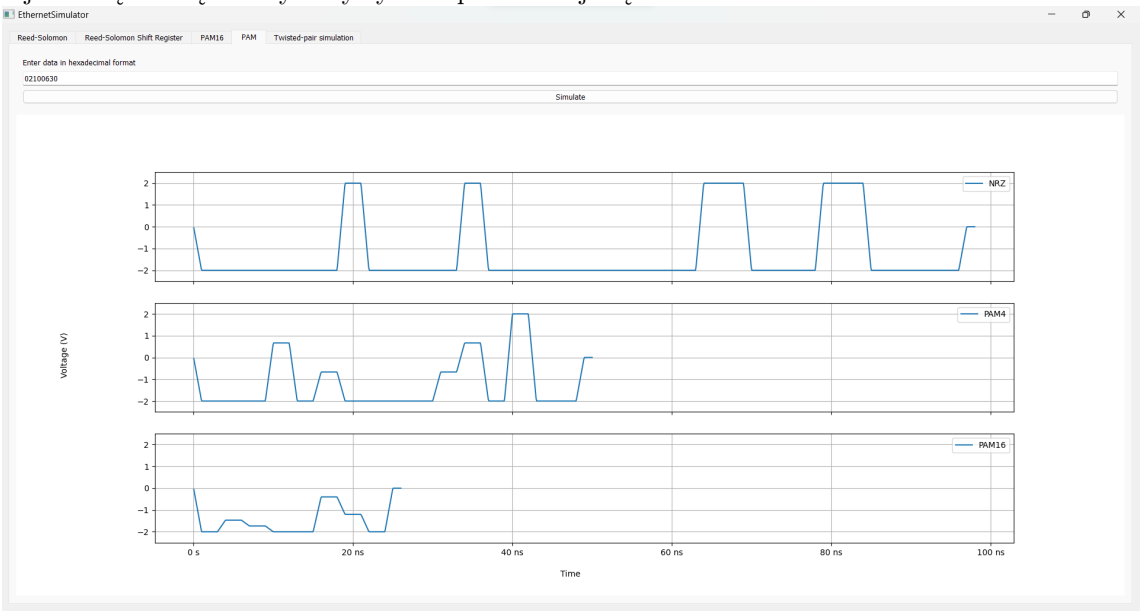
Rysunek 18: Wykres symulacji stworzony przy użyciu Matplotlib

Najprostszym schematem modulacji jest NRZ — Non-Return-to-Zero. Chcąc przesłać bit o wartości 0, na skrętkę zostanie podany sygnał ujemny, a w przypadku 1 — dodatni. Rozwiązanie te niesie za sobą parę wad. Przykładowo, gdy nadawane są długie ciągi zer lub jedynek, sygnał nie ulega zmianie — jest to zjawisko niepożądane podczas transmisji i może doprowadzić do desynchronizacji zegarów strony nadawczej i odbiorczej. Z uwagi na to, NRZ wykorzystywany jest w praktyce w połączeniu z np. kodowaniem liniowym 64b/66b w celu uniknięcia sekwencji zer i jedynek.

PAM jest modulacją, w której dane przesyłane są w postaci zmian amplitudy sygnału. Zmiany te nazywane symbolami. Modulacje PAM różnią się między sobą liczbą wykorzystywanych poziomów modulacji. PAM3 wykorzystuje trzy poziomy, PAM4 — cztery, PAM16 — szesnaście. NRZ można wobec tego nazwać PAM2. Powodem, dla którego zwiększenie poziomów ma sens jest zwiększona szybkość transmisji. Weźmy na przykład PAM4 — mając cztery poziomy mamy do dyspozycji cztery symbole -3, -1, 1, 3, a więc każdy symbol kodować może dwa bity danych. W przypadku NRZ, jeden symbol koduje tylko jeden bit. Zatem zwiększenie liczby poziomów pozwala na przesył większej ilości bitów przy użyciu jednego symbolu. Schemat ten będzie działał, o ile strona odbiorcza potrafi rozróżnić poszczególne symbole od siebie, jest to jednak łatwiej osiągalne niż zwiększenie szerokości pasma. Modulacje PAM4, PAM16 i inne, analogicznie jak NRZ, podatne są na długie ciągi zer i jedynek. Dlatego niezbędne jest zastosowanie różnych metod zamieniających takie sekwencje na bardziej zróżnicowany ciąg np. skrambler.

Opis narzędzia

Program phyether, w zakładce ‘PAM‘, posiada symulator modulacji NRZ, PAM4 oraz PAM16, który ilustruje zachowanie sygnału w skrętce podczas transmisji, w zależności od wybranych modulacji. Narzędzie będzie wykorzystywane podczas tej części ćwiczenia.



Powyższa grafika prezentuje zakładkę ‘PAM‘ programu phyether. Nad przyciskiem **Simulate** znajduje się pole tekstowe, do którego wpisane mogą być dane w formie liczb w systemie szesnastkowym. Po wpisaniu danych i kliknięciu **Simulate**, wpisane dane zamieniane są na symbole modulacji NRZ, PAM4 oraz PAM16, które następnie wysyłane są na medium. Wynik symulacji przedstawiony jest w dolnej części zakładki.

Zadania do realizacji

1. Zamień numer swojego indeksu na postać szesnastkową i wykorzystaj go jako liczbę do przesłania. Przeprowadź symulację. Zanotuj w sprawozdaniu przybliżony czas transmisji oraz liczbę poziomów natężenia. Opisz wnioski, które nasuwają Ci się po wykonanym ćwiczeniu.

2. Prześlij ciąg składający się z samych jedynek (ffffff ...) lub samych zer (000000 ...). Popatrz na wynik symulacji. Jak nazywa się zaobserwowane zjawisko? Czy znasz sposoby, które zapobiegają jego wystąpieniu? Zanotuj w sprawozdaniu.

Ćwiczenie dydaktyczne — PAM16 w 40GBASE-T

Wstęp teoretyczny

Ciekawą kwestią w przypadku modulacji PAM jest także zamiana bitów na konkretne symbole alfabetu modulacji. Możemy sobie wyobrazić, że w najprostszym przypadku wartość 0 mapowana jest na najmniejszy symbol, natomiast 1111 — na największy. Istnieją także bardziej wyrafinowane

sposoby mapowania, przykładowo popularnym jest zastosowanie kodu Graya. W tym ćwiczeniu zobaczymy podejście, które zostało zastosowane w standardach 25GBASE-T oraz 40GBASE-T.

Wyżej wymienione standardy korzystają z modulacji PAM16, natomiast symbole nadawane na parach skrętki wybierane są według diagramu konstelacji DSQ128. Aby wyjaśnić czym on jest, spójrzmy pierw na 64QAM (Quadrature Amplitude Modulation). 64QAM to modulacja, która jest połączeniem modulacji amplitudy oraz fazy (dane kodowane są poprzez zmianę amplitudy oraz fazy sygnału). Symbole wybierane są na podstawie diagramu konstelacji 64QAM — rys. 19.

Diagram podzielony jest na cztery części. Oś X odnosi się do fazy, a oś Y — amplitudy. Każdy symbol, który chcemy nadać, jest krotką (x, y), która oznacza punkt na diagramie dla danej wartości. Jak można zauwarzyć, sąsiednie punkty różnią się między sobą jednym bitem — dzięki czemu, gdy nastąpi przekłamanie, najprawdopodobniej tylko jeden bit będzie zły.

W standardach 25GBASE-T i 40GBASE-T symbole dobierane są na podstawie diagramu konstelacji DSQ128, który jest złożeniem dwóch diagramów 64QAM - rys. 20.

Dane grupowane są w 7-bitowe grupy (u_0, u_1, u_2), (c_0, c_1, c_2, c_3), po czym każda taka grupa mapowana jest na krotkę (PAM16_1, PAM16_2) według algorytmu:

Krok 1:

$$x_{13} = \neg u_0 * u_2$$
$$x_{12} = u_0 \oplus u_2$$
$$x_{11} = c_0$$
$$x_{10} = c_0 \oplus c_1$$
$$x_{23} = (u_1 * u_2) + (u_0 * \neg u_1)$$
$$x_{22} = u_1 \oplus u_2$$
$$x_{21} = c_2$$
$$x_{20} = c_2 \oplus c_3$$

Krok 2:

$$x_1 = 8x_{13} + 4x_{12} + 2x_{11} + x_{10}$$
$$x_2 = 8x_{23} + 4x_{22} + 2x_{21} + x_{20}$$

Krok 3:

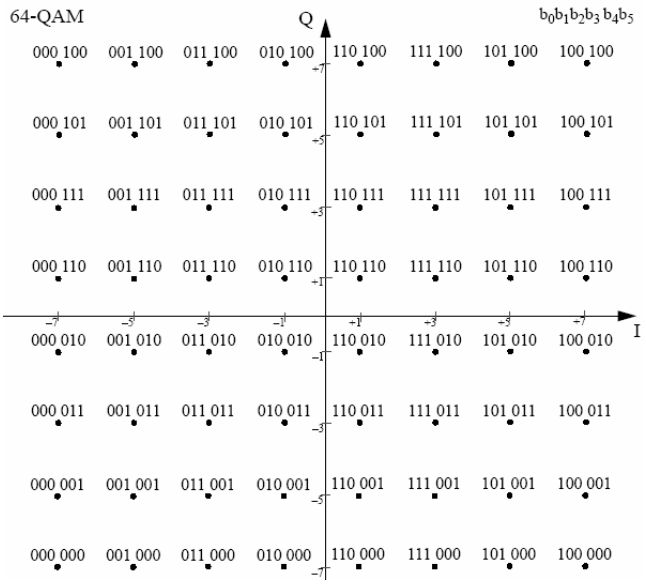
$$y_1 = (x_1 + x_2) \mod 16$$
$$y_2 = (-x_1 + x_2) \mod 16$$

Krok 4:

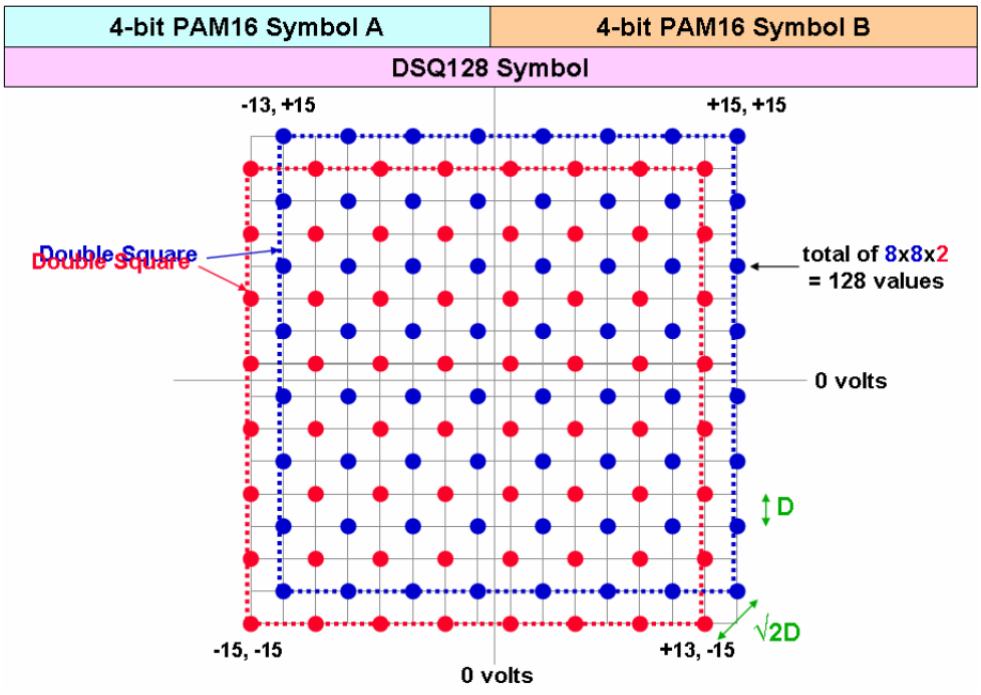
$$\text{PAM16}_1 = 2y_1 - 15$$
$$\text{PAM16}_2 = 2y_2 - 15$$

Symbole są następnie nadawane na kolejnych parach skrętki:

Pair A	$PAM16_1<0>$	$PAM16_2<0>$	$PAM16_1<4>$	$PAM16_2<4>$	...	$PAM16_1<508>$	$PAM16_2<508>$
Pair B	$PAM16_1<1>$	$PAM16_2<1>$	$PAM16_1<5>$	$PAM16_2<5>$	...	$PAM16_1<509>$	$PAM16_2<509>$
Pair C	$PAM16_1<2>$	$PAM16_2<2>$	$PAM16_1<6>$	$PAM16_2<6>$	...	$PAM16_1<510>$	$PAM16_2<510>$
Pair D	$PAM16_1<3>$	$PAM16_2<3>$	$PAM16_1<7>$	$PAM16_2<7>$	...	$PAM16_1<511>$	$PAM16_2<511>$



Rysunek 19: Diagram konstelacji 64QAM



Rysunek 20: Diagram konstelacji DSQ128

Zadania do realizacji

1. Zastosowanie DSQ128 nie eliminuje możliwości wystąpienia stałej składowej. Znajdź ciąg, który temu dowodzi i zanotuj go w sprawozdaniu.

Ćwiczenie dydaktyczne — Kodowanie Reeda-Solomona

Wstęp teoretyczny

Kodowanie korekcyjne Reeda-Solomona zostało stworzone przez Irvina S. Reeda oraz Gustava Solomona w 1960 roku. Kody Reeda-Solomona charakteryzują się kilkoma parametrami:

- alfabetem w ciele skończonym $\mathbb{F}_{2^m}$, $m > 1$
- długością wiadomości k do zakodowania $k < 2^m$
- długością słowa kodowego n gdzie $k < n < 2^m$
- wielomianem generującym $g(x)$

Kody Reeda-Solomona cechują się możliwością korekty $\lfloor \frac{n-k}{2} \rfloor$ lub wykrycia $n-k$ błędnych symboli. Symbol w ciele $\mathbb{F}_{2^m}$ składa się z m bitów co w przypadku błędów grupowych daje możliwość korekty maksymalnie $m \cdot \lfloor \frac{n-k}{2} \rfloor$ bitów bądź detekcji $m(n-k)$ przekłamanych bitów

Aby zrozumieć działanie kodu Reeda-Solomona trzeba najpierw zrozumieć czym jest ciało skończone $\mathbb{F}_q$ zwane też ciałem Galois $\mathrm{GF}(q)$.

Ciało skończone $\mathbb{F}_q$

Ciało to jest ciałem K rzędu q czyli takie które zawiera jedynie q elementów. Aby ciało skończone istniało q musi spełniać warunek $q = p^k$, $k \in \{1, 2, \ldots\}$ gdzie p jest liczbą pierwszą oraz definiować działania dodawania i mnożenia spełniające kilka warunków:

- dodawanie i mnożenie jest łączne, przemienne oraz zawiera elementy neutralne
- każdy element musi posiadać element odwrotny względem dodawania
- każdy element oprócz 0 musi posiadać element odwrotny względem mnożenia
- mnożenie jest rozdzielne względem dodawania

Aby stworzyć ciało $\mathbb{F}_p$ gdzie p jest liczbą pierwszą można wykorzystać pierścień klas reszt $\mathbb{Z}/p\mathbb{Z}$ w którym działania to zwykłe dodawanie i mnożenie modulo

$$\mathbb{Z}/p\mathbb{Z} = \{[a]_p \mid a \in \mathbb{Z}\} = \{[0]_p, [1]_p, [2]_p, \ldots, [p-1]_p\}$$
$$[a]_p + [b]_p = [a+b]_p$$
$$[a]_p \cdot [b]_p = [a \cdot b]_p$$

W tym wprowadzeniu będą używane jedynie ciała $\mathbb{F}_2$ oraz $\mathbb{F}_{2^m}$ z których korzysta kod Reeda-Solomona.

Ciało $\mathbb{F}_2$

Ciało $\mathbb{F}_2$ jest jednym z najczęściej używanych ciał w informatyce. Ciało to definiuje 2 elementy $\{0, 1\}$ w którym działania + i $\cdot$ są równoważne operacjom logicznym XOR oraz AND

Dodawanie i mnożenie w $\mathbb{F}_2$

a	b	+	$\cdot$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ciało $\mathbb{F}_{2^m}$

Aby zdefiniować ciało skończone $\mathbb{F}_{2^m}$ potrzebujemy najpierw znaleźć wielomian nierozkładalny $p(x)$ stopnia m, czyli taki który nie da się przedstawić jako iloczyn dwóch innych wielomianów stopnia mniejszego $p(x) = g(x) \cdot f(x)$

Wielomian jest ten potrzebny do zdefiniowania działania mnożenia ponieważ elementy tego ciała interpretowane są jako wielomiany w postaci:

$$\sum_{n=0}^{m-1} c_n \alpha^n = c_0 + c_1\alpha + c_2\alpha^2 + \cdots + c_{m-1}\alpha^{m-1}, c_n \in \mathbb{F}_2$$

Wielomiany te mogą być reprezentowane jako liczby binarne $c_{m-1}c_{m-2}\ldots c_1c_0$ bądź po konwersji liczby binarnej jako np. liczby dziesiętne. Przykładowe wielomiany w ciele $\mathbb{F}_{2^4}$ i ich różne zapisy podane są w tabelce:

Interpretacje niektórych elementów ciała $\mathbb{F}_{2^4}$

Wielomian	Liczba Binarna	Liczba dziesiętna
$x^3 + x^2 + x + 1$	1111	15
$x^3 + x$	1010	9
$x + 1$	11	3

Dodawanie elementów w ciele $\mathbb{F}_{2^m}$ jest po prostu zwykłym dodawaniem wielomianów, trzeba tylko pamiętać, że współczynniki dodajemy w ciele $\mathbb{F}_2$ czyli XORujemy:

$$\sum_{n=0}^{m-1} c_n \alpha^n + \sum_{n=0}^{m-1} d_n \alpha^n = \sum_{n=0}^{m-1} (c_n + d_n)\alpha^n$$

Wynikiem mnożenia a i b w ciele $\mathbb{F}_{2^m}$ jest reszta z dzielenia iloczynu tych wielomianów przez wielomian nierozkładalny $p(x)$. Dzielenie jak i mnożenie można wykonać tak jak na zwykłych wielomianach, pamiętając o tym, że współczynniki są w ciele $\mathbb{F}_2$.
Przykładowe działanie mnożenia w $\mathbb{F}_{2^2}$, wielomian nierozkładalny $p(x) = x^2 + x + 1$

$$
\begin{aligned}
a = \alpha + 1,\ b = \alpha + 1 & \\
a \cdot b = (\alpha + 1)^2 & \mod x^2 + x + 1 \\
a \cdot b = \alpha^2 + [2]_2 x + 1 & \mod x^2 + x + 1 \\
a \cdot b = \alpha^2 + 1 & \mod x^2 + x + 1 \\
a \cdot b = \alpha &
\end{aligned}
$$

Zamiast liczenia reszty z dzielenia można także skorzystać z pewnej zależności. Zdefiniujmy α jako pierwiastek wielomianu $p(x)$.

$$
\begin{aligned}
[-1]_2 &\equiv [1]_2 \\
p(\alpha) &= 0 \\
\alpha^2 + \alpha + 1 &= 0 \\
\alpha^2 &= -\alpha - 1 \\
\alpha^2 &= \alpha + 1
\end{aligned}
$$

Teraz zamiast liczyć resztę z dzielenia możemy po prostu podstawić α^2:

$$
\begin{aligned}
a \cdot b &= \alpha^2 + 1 \\
a \cdot b &= (\alpha + 1) + 1 \\
a \cdot b &= \alpha
\end{aligned}
$$

Oryginalny kod Reeda-Solomona

Sposób kodowania przedstawiony w pracy Reeda i Solomona polega na stworzeniu wielomianu $p_m(x) = \sum_{i=0}^{k-1} m_i x^i$, gdzie $m_i \in \mathbb{F}_{2^m}$ to i-ty element wiadomości, po czym za pomocą tego wielomianu obliczane jest słowo kodowe $C(m) = (p_m(a_0), p_m(a_1), \ldots, p_m(a_{n-1}))$ gdzie a_i to różne elementy ciała $\mathbb{F}_{2^m}$.
Przykładowy kod RS dla $\mathbb{F}_{2^2}$, $p(x) = x^2 + x + 1$, $n = 3$, $k = 2$, wiadomość to 2 liczby 2 i 3.

$$p_m(x) = 3x + 2$$
$$p_m(0) = 3 \cdot 0 + 2 = 2$$
$$p_m(1) = 3 \cdot 1 + 2 = 3 + 2 = 0\text{b}11 + 0\text{b}10 = 0\text{b}01 = 1$$
$$p_m(2) = 3 \cdot 2 + 2 = (\alpha + 1) \cdot \alpha + \alpha$$
$$p_m(2) = 3 \cdot 2 + 2 = \alpha^2 + \alpha + \alpha = \alpha + 1 = 3$$
$$p_m(3) = 3 \cdot 3 + 2 = (\alpha + 1)^2 + \alpha = \alpha + \alpha = 0$$

Zakodowana wiadomość to "2 1 3 0"

Kod systematyczny

Za pomocą niewielkiej modyfikacji można stworzyć kod systematyczny czyli taki w którym słowo kodowe zawiera w sobie kodowaną wiadomość. Żeby stworzyć kod systematyczny musimy zmodyfikować sposób tworzenia wielomianu w taki sposób by $p_m(x_i) = m_i$ dla $i \in \{0, 1, \ldots, k-1\}$.

Jednym ze sposobów stworzenia takiego wielomianu jest użycie metody interpolacji wielomianów. Słowo kodowe wygenerowane z tego wielomianu będzie zawierało wiadomość w pierwszych k elementach.

$$C(m) = (p_m(a_0), p_m(a_1), \ldots, p_m(a_{n-1})) = (m_0, m_1, \ldots, m_{k-1}, p_m(a_k), p_m(a_{k+1}), \ldots, p_m(a_{n-1}))$$

kod BCH

Kody BCH (Bose-Chaudhuri-Hocquenghem) są kodami cyklicznymi co oznacza że każde przesunięcie słowa kodowego jest także słowem kodowym np.

$$(c_0, c_1, \ldots, c_{n-2}, c_{n-1}),\ (c_{n-1}, c_0, \ldots, c_{n-3}, c_{n-2})$$

Aby zbudować kod BCH Reeda-Solomona potrzebujemy najpierw funkcji minimalnej pierwiastka α, czyli takiego minimalnego wielomianu nierozkładalnego $p(x)$ stopnia m dla którego istnieje element prymitywny (pierwiastek) α który pozwala wygenerować całe ciało skończone

$$\mathbb{F}_{2^m} = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{p^m - 1}\}$$

Mając taki element prymitywny jesteśmy w stanie stworzyć wielomian generujący $g(x)$ używając wzoru

$$t = n - k$$
$$g(x) = \prod_{i=0}^{t-1} (x - \alpha^i) = g_t x^t + g_{t-1} x^{t-1} + \cdots + g_1 x + g_0$$

Aby utworzyć słowo kodowe wystarczy pomnożyć wielomian $p_m(x)$ przez wielomian generujący $g(x)$

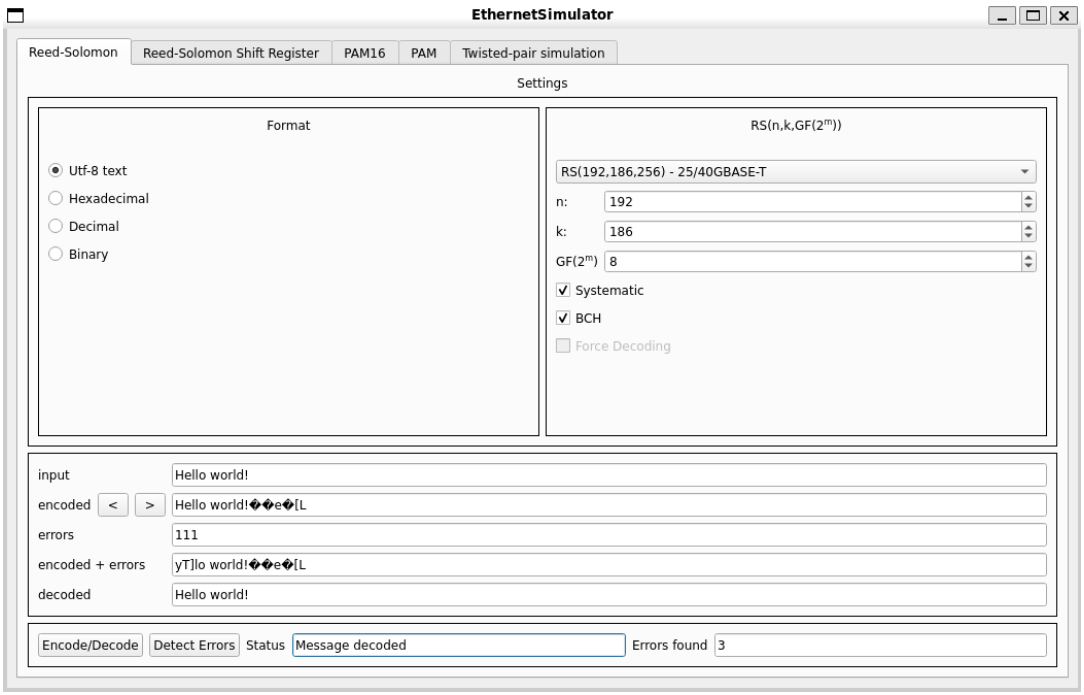
systematyczny kod BCH

Aby uzyskać systematyczne słowo kodowe $s(x)$ musimy obliczyć:

$$s_r(x) = p_m(x) \cdot x^t \mod g(x)$$
$$s(x) = p_m(x) \cdot x^t - s_r(x)$$

Narzędzia

Reed-Solomon



Narzędzie pozwala na:

- Kodowanie i dekodowanie wiadomości różnymi koderami o różnych parametrach
- Nakładanie na zakodowaną wiadomość błędów w celu przetestowania możliwości detekcji i korekty błędów przez kod Reeda-Solomona
- Zmianę formatu symboli w celu łatwiejszego zobrazowania co się dzieje w poszczególnych etapach

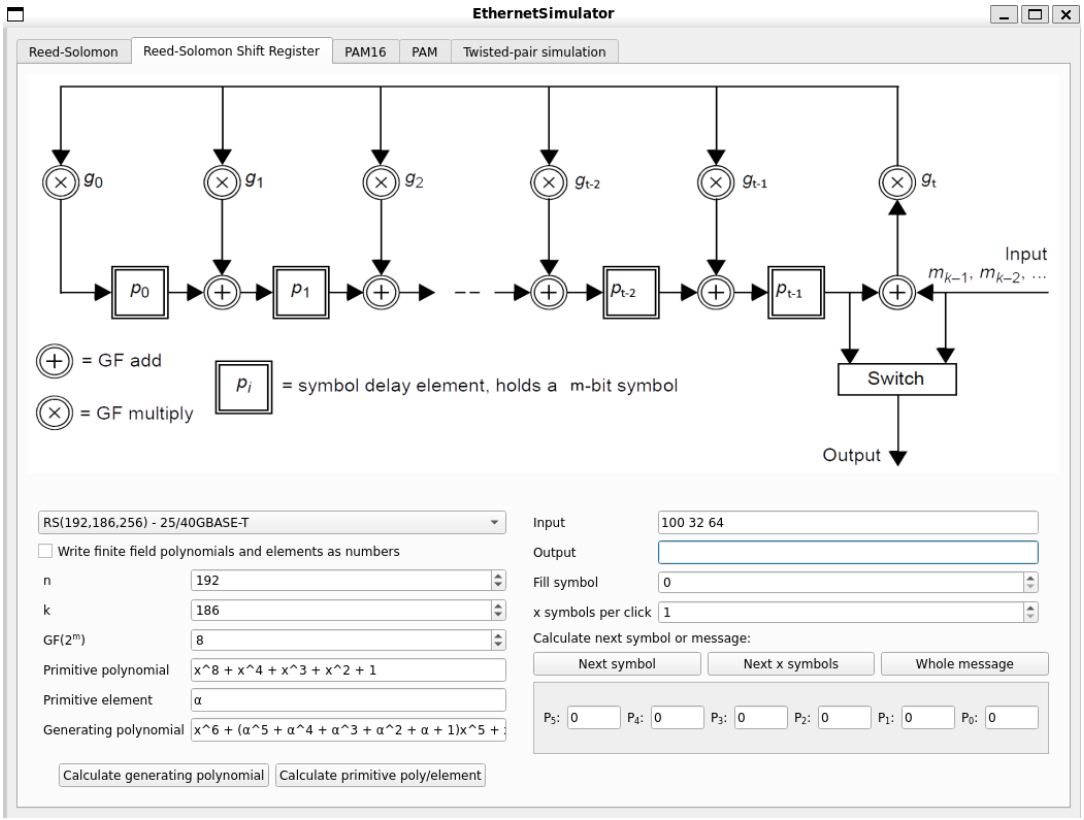
Zakładka Reed-Solomon składa się z 3 części

- Format — pozwala na zmianę formatu wyświetlanych danych. Tryb tekstowy pozwala na szybkie zorientowanie się jak dane się zmieniają. Tryb binarny pozwoli na dokładne dodanie błędów do słowa kodowego. Kodowanie znaków tekstowych to UTF-8 w związku z czym polskie znaki diakrytyczne kodowane są na 2 bajtach.

- RS(n, k, $GF(2^m)$) — parametry kodu RS (wielomian $p(x)$ jest obliczany automatycznie).
- Dane — tutaj możemy wpisać wiadomość którą chcemy zakodować lub błędy które będą XORowane ze słowem kodowym oraz zobaczyć wynik kodowania. Status informuje nas o tym czy znaleziono błędy bądź czy udało się zdekodować wiadomość. 'Errors found' informuje ile błędów zostało poprawionych.

Tryb tekstowy nie jest w stanie poprawnie wyświetlić wszystkich znaków, część jest zamieniana na znaki zapytania a część jest tzw. znakami białymi (spacje, tabulatory bądź znaki niewyświetlane). Strzałki przy 'encoded' pozwalają na przesunięcie symboli w lewo i prawo.

Reed-Solomon Shift Register



Narzędzie pozwala na:

- Obliczanie wielomianu generującego kod Reeda-Solomona
- Zobrazowanie jak działa przykładowy koder Reeda-Solomona
- Zaznajomienie się z reprezentacją w postaci wielomianowej symboli z ciała $\mathbb{F}_{2^m}$

W tej zakładce koder RS zaimplementowany został zgodnie z modelem funkcyjnym udostępnionym w standarcie Ethernet. Po przejściu wszystkich symboli wiadomości element 'Switch' zacznie przepuszczać symbole parzystości. W opcjach po lewej możemy podobnie jak w poprzedniej zakładce wybrać parametry kodera oraz dodatkowo wybrać inne wielomiany i elementy prymitywne.

Przycisk ‘Calculate generating polynomial’ obliczy wielomian generujący a ‘Calculate primitive poly/ement’ obliczy element i wielomian prymitywny dla podanego ciała skończonego $\mathbb{F}_{2^m}$. Po prawej stronie mamy dane wejściowe oraz aktualny stan rejestrów p_i. ‘Fill symbol’ jest symbolem który będzie wysyłany jeżeli zabraknie symboli na wejściu.

Zadania

- Zakładka: Reed-Solomon, ustawienia programu: kod systematyczny BCH, $n = 15$, $GF = 2^4$. Dla $k \in \{2, 6, 10, 13\}$ sprawdź wartość słowa kodowego dla k-symbolowej wiadomości zawierającej same zera. Dlaczego otrzymałeś takie słowa kodowe?

- Zakładka: Reed-Solomon, ustawienia programu: $n = 7$, $k = 3$, $GF = 2^3$, kod systematyczny BCH, tryb dziesiętny, wiadomość wejściowa ‘1 2 3’. Sprawdź dla błędów ‘3’, ‘3 2’, ‘3 2 1’ oraz ‘3 2 1 4’ czy dekoder jest w stanie wykryć błędy oraz czy jest w stanie je poprawić a jeżeli tak to czy poprawnie. Czy wiesz dlaczego pojawiły się rozbieżności między błędami poprawionymi a wykrytymi?

- Zakładka: Reed-Solomon Shift Register, ustawienia programu: $n = 3$, $k = 2$, $GF = 2^2$. Oblicz wielomiany prymitywne naciskając przycisk ‘Calculate primitive poly/element’. Zakoduj wiadomość: ‘2 1’ w symulatorze po czym zakoduj wiadomość używając wzoru z sekcji ‘Systematyczny kod BCH’. Porównaj wyniki.

- Zakładka: Reed-Solomon, ustawienia programu: $n = 15$, $k = 7$, $GF = 2^4$, kod systematyczny BCH. Zakoduj dowolną niezerową k-symbolową wiadomość. Sprawdź czy przesunięcia słowa kodowego (z użyciem strzałek przy słowie ‘encoded’) także będą słowem kodowym.

Dodatek B: Wejściówka

1. Ile błędnych symboli jest w stanie wykryć lub poprawić kod Reeda-Solomona?

   A) wykryć: $n-k$, poprawić: $n-k-1$

   B) wykryć: $n-k-1$, poprawić: $n-k-1$

   C) wykryć: $\lfloor \frac{n-k}{2} \rfloor$, poprawić: $\lfloor \frac{n-k}{2} \rfloor$

   D) wykryć: $n-k$, poprawić: $\lfloor \frac{n-k}{2} \rfloor$

2. Podaj zaletę oraz wadę stosowania większej ilości poziomów w modulacjach PAM.

3. Opisz krótko czym jest NRZ (Non-Return-to-Zero).

4. Podaj definicję dodawania i mnożenia w $\mathbb{F}_2$ bądź wypisz wynik tych działań dla wszystkich możliwych kombinacji elementów.

5. Czym się różni słowo kodowe wygenerowane kodem systematycznym i niesystematycznym?

Dodatek C: Zadania laboratoryjne

1. Zakładka: Reed-Solomon, ustawienia programu: kod systematyczny BCH, $n = 15$, $GF = 2^4$. Dla $k \in \{2, 6, 10, 13\}$ sprawdź wartość słowa kodowego dla k-symbolowej wiadomości zawierającej same zera. Dlaczego otrzymałeś takie słowa kodowe?

2. Zakładka: Reed-Solomon, ustawienia programu: $n = 7$, $k = 3$, $GF = 2^3$, kod systematyczny BCH, tryb dziesiętny, wiadomość wejściowa '1 2 3'. Sprawdź dla błędów '3', '3 2', '3 2 1' oraz '3 2 1 4' czy dekoder jest w stanie wykryć błędy oraz czy jest w stanie je poprawić a jeżeli tak to czy poprawnie. Czy wiesz dlaczego pojawiły się rozbieżności między błędami poprawionymi a wykrytymi?

Błąd	**Wykrywa [TAK/NIE]**	**Poprawia [Ile]**	**Poprawia poprawnie [TAK/NIE]**
3			
3 2			
3 2 1			
3 2 1 4			

3. Zakładka: Reed-Solomon Shift Register, ustawienia programu: $n = 3$, $k = 2$, $GF = 2^2$. Oblicz wielomian prymitywny i generator naciskając przycisk 'Calculate primitive poly/element'. Zakoduj wiadomość: '2 1' w symulatorze po czym zakoduj wiadomość używając wzoru z sekcji 'Systematyczny kod BCH'. Porównaj wyniki.

4. Zakładka: Reed-Solomon, ustawienia programu: $n = 15$, $k = 7$, $GF = 2^4$, kod systematyczny BCH. Zakoduj dowolną niezerową k-symbolową wiadomość. Sprawdź czy przesunięcia słowa kodowego (z użyciem strzałek przy słowie 'encoded') także będą słowem kodowym.

5. Zamień numer swojego indeksu na postać szesnastkową i wykorzystaj go jako liczbę do przesłania. Przeprowadź symulację. Zanotuj w sprawozdaniu przybliżony czas transmisji oraz liczbę poziomów natężenia. Opisz wnioski, które nasuwają Ci się po wykonanym ćwiczeniu.

6. Prześlij ciąg składający się z samych jedynek (fffffff ...). Popatrz na wynik symulacji. Jak nazywa się zaobserwowane zjawisko? Czy znasz sposoby, które zapobiegają jego wystąpieniu? Zanotuj w sprawozdaniu.

Dodatek D: Kod źródłowy symulatora

Kod źródłowy programu dostępny jest na załączonym urządzeniu przenośnym oraz pod adresem: https://github.com/iwanicki92/Ethernet-physical-layer-simulator/tree/dev.

Dodatek E: Instrukcja instalacji symulatora

Z dostępem do Internetu

W przypadku dostępu do Internetu wystarczy zainstalować wersję języka Python spełniającą warunek: $3.9 \leq$ wersja ≤ 3.11 oraz uruchomić polecenie instalacji symulatora:
pip install git+https://github.com/iwanicki92/Ethernet-physical-layer-simulator.git@dev

Bez dostępu do Internetu

W przypadku braku dostępu do Internetu konieczne jest skopiowanie zawartości specjalnie przygotowanego do tej okazji urządzenia przenośnego dołączonego do pracy. Zawiera on gotowe środowisko pythonowe oraz kod źródłowy symulatora.

Dodatek F: Rozwiązania zadań

Rozwiązania wejściówki

1. Podaj definicję dodawania i mnożenia w $\mathbb{F}_2$ bądź wypisz wynik tych działań dla wszystkich możliwych kombinacji elementów.
    - Odpowiedź: dodawanie to XOR mnożenie to AND. Pozostałe możliwe odpowiedzi to Tablica 1 bądź (11) oraz (12)
2. Czym się różni słowo kodowe wygenerowane kodem systematycznym i niesystematycznym
    - Odpowiedź: słowo kodowe w kodowaniu systematycznym w przeciwieństwie do niesystematycznego zawiera w sobie kodowaną wiadomość, sekcja 2.6.2
3. Ile błędnych symboli jest w stanie wykryć lub poprawić kod Reeda-Solomona?
    - Odpowiedź wykryć: $n - k$, poprawić: $\lfloor \frac{n-k}{2} \rfloor$, sekcja 2.5
4. Podaj zaletę oraz wadę stosowania większej ilości poziomów w modulacjach PAM.
5. Opisz krótko czym jest NRZ (Non-Return-to-Zero).

Rozwiązania ćwiczeń

1. Zakładka: Reed-Solomon, ustawienia programu: kod systematyczny BCH, $n = 15$, $GF = 2^4$. Dla $k \in \{2, 6, 10, 13\}$ sprawdź wartość słowa kodowego dla k-symbolowej wiadomości zawierającej same zera. Dlaczego otrzymałeś takie słowa kodowe?

   Odpowiedź: Otrzymaliśmy same zera ponieważ według wzoru z sekcji 2.6.4 dla $p_m(x) = 0$ zawsze dostaniemy zera

   $$c_r(x) = 0 \cdot x^t \mod g(x)$$
   $$c_r(x) = 0 \mod g(x)$$
   $$c_r(x) = 0$$
   $$c(x) = 0 \cdot x^t - 0$$
   $$c(x) = 0$$

2. Zakładka: Reed-Solomon, ustawienia programu: $n = 7$, $k = 3$, $GF = 2^3$, kod systematyczny BCH, tryb dziesiętny, wiadomość wejściowa '1 2 3'. Sprawdź dla błędów '3', '3 2', '3 2 1' oraz '3 2 1 4' czy dekoder jest w stanie wykryć błędy oraz czy jest w stanie je poprawić a jeżeli tak to czy poprawnie. Czy wiesz dlaczego pojawiły się rozbieżności między błędami poprawionymi a wykrytymi?

   Odpowiedź: Według właściwości kodu Reeda-Solomona opisanych w sekcji 2.5 jest on w stanie poprawić $\lfloor \frac{n-k}{2} \rfloor$ błędnych symboli i wykryć $n - k$ błędnych symboli. W tym zadaniu wszystkie błędy były wykryte i jedynie $\lfloor \frac{n-k}{2} \rfloor$ czyli 2 błędy były poprawnie skorygowane.

Błąd	**Wykrywa [TAK/NIE]**	**Poprawia [Ile]**	**Poprawia poprawnie [TAK/NIE]**
3	TAK	1	TAK
3 2	TAK	2	TAK
3 2 1	TAK	2	NIE
3 2 1 4	TAK	2	NIE

3. Zakładka: Reed-Solomon Shift Register, ustawienia programu: $n = 3$, $k = 2$, $GF = 2^2$. Oblicz wielomian prymitywny i generator naciskając przycisk 'Calculate primitive poly/element'. Zakoduj wiadomość: '2 1' w symulatorze po czym zakoduj wiadomość używając wzoru z sekcji

‘Systematyczny kod BCH’. Porównaj wyniki.

Odpowiedź: Zakodowana wiadomość to: ‘2 1 3’. Dla $p_m(x) = 2x + 1$ oraz $g(x) = x + 1$ słowo kodowe obliczone wzorem i przez symulator są takie same.

$$
\begin{aligned}
s_r(x) &= (2x+1) \cdot x \mod x+1 \\
s_r(x) &= 2x^2 + x \mod x+1 \\
s_r(x) &= 3 \\
s(x) &= (2x+1) \cdot x - s_r(x) \\
s(x) &= 2x^2 + x + 3 \\
s(x) &= (2, 1, 3)
\end{aligned}
$$

4. Zakładka: Reed-Solomon, ustawienia programu: $n = 15$, $k = 7$, $GF = 2^4$, kod systematyczny BCH. Zakoduj dowolną niezerową k-symbolową wiadomość. Sprawdź czy przesunięcia słowa kodowego (z użyciem strzałek przy słowie ‘encoded’) także będą słowem kodowym.

   Odpowiedź: Dla wiadomości ‘1 2 3 4 5 6 7’ wszystkie przesunięcia były poprawnie i bezbłędnie dekodowane.

5. Zakładka: PAM, Zamień numer swojego indeksu na postać szesnastkową i wykorzystaj go jako liczbę do przesłania. Przeprowadź symulację. Zanotuj w sprawozdaniu przybliżony czas transmisji oraz liczbę poziomów natężenia. Opisz wnioski, które nasuwają Ci się po wykonanym ćwiczeniu.

   Odpowiedź:

   - NRZ: 62ns, dwa poziomy
   - PAM4: 33ns, cztery poziomy
   - PAM16: 17ns, cztery poziomy (przesyłany numer indeksu jest na tyle mały, że nie wszystkie poziomy zostaną wykorzystane)

   Stosowanie modulacji PAM wyższych poziomów pozwala na lepsze upakowanie danych, ale różnica pomiędzy wysyłanymi symbolami maleje.

6. Zakładka: PAM, Prześlij ciąg składający się z samych jedynek (fffffff ...) lub zer (000000 ...). Popatrz na wynik symulacji. Jak nazywa się zaobserwowane zjawisko? Czy znasz sposoby, które zapobiegają jego wystąpieniu? Zanotuj w sprawozdaniu.

   Odpowiedź: Na symulacji widać stałą składową. Można jej zapobiegać używając np. kodowania liniowego 64b/66b albo skramblera.

7. Zakładka: PAM16, Zastosowanie DSQ128 nie eliminuje możliwości wystąpienia stałej składowej. Znajdź ciąg, który temu dowodzi i zanotuj go w sprawozdaniu.

   Odpowiedź: Na przykład: 0x000000 ..., 0x111111 ... (0b000100010001 ...), 0x22222 ... (0b001000100010 ...), 0x444444 ... (0b010001000100 ...).