Dydaktyczny symulator wybranych rozwiązań warstwy fizycznej sieci Ethernet

Imię i nazwisko studenta: **Michał Iwanicki**

Poziom kształcenia: **Stacjonarne**

Kierunek studiów: **Informatyka**

Profil: **Algorytmy i modelowanie systemów**

Imię i nazwisko studenta: **Mateusz Bauer**

Poziom kształcenia: **Stacjonarne**

Kierunek studiów: **Informatyka**

Profil: **Teleinformatyka**

Imię i nazwisko studenta: **Marcin Garnowski**

Poziom kształcenia: **Stacjonarne**

Kierunek studiów: **Informatyka**

Profil: **Inteligentne Systemy Interaktywne**

Promotor pracy: **dr. inż. Krzysztof Nowicki**

Spis treści

1 Środowisko programistyczne

1.1 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.

1.2 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, na które się zdecydowano:

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.

2 Symulator

2.1 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.

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.

2.2 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 ułatwienie zrozumienia działania tych rozwiązań.

2.2.1 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.

2.2.2 Symulacja przesyłu sygnału

Symulator umożliwia symulację przesyłu danych przez skrętkę lub kablem Ethernet, w której można śledzić zmiany napięć. W tym przypadku wykorzystana została biblioteka PySpice. Dzięki niej można nadać wykorzystywanemu przewodowi pożądane parametry, między innymi: opór, długość i indukcyjność.

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

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.

3 Kodowanie korekcyjne Reeda-Solomona

3.1 Wstęp

Kodowanie korekcyjne Reeda-Solomona zostało stworzone przez Irvina S. Reeda oraz Gustava Solomona w 1960 roku [1] Kody Reeda-Solomona charakteryzują się 3 parametrami, rozmiarem alfabetu q [2] interpretowanym w ciele skończonym $\mathbb{F}_q$, długością wiadomości do zakodowania k oraz długością słowa kodowego n gdzie $k < n \leq q$ oraz $q = p^m$ gdzie p to liczba pierwsza a $m \in \{2, 3, \ldots\}$

3.2 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)
$a * (b * c) = (a * b) * c$	$\forall a, b, c \in K$	Łączność mnożenia	(2)
$a + b = b + a$	$\forall a, b \in K$	Przemienność dodawania	(3)
$a * b = b * a$	$\forall a, b \in K$	Przemienność mnożenia	(4)
$a + 0 = a$	$\forall a \in K$	Element neutralny (0) dodawania	(5)
$a * 1 = a$	$\forall a \in K$	Element neutralny (1) mnożenia	(6)
$a + (-a) = 0$	$\forall a \in K$	Element odwrotny (-a) dodawania	(7)
$a * a^{-1} = 1$	$\forall a \in K \setminus \{0\}$	Element odwrotny (a^{-1}) mnożenia	(8)
$a * (b + c) = (a * b) + (a * c)$	$\forall a, b, c \in K$	Rozdzielność mnożenia względem dodawania	(9)

W kodzie solomona wykorzystujemy ciało $\mathbb{F}_2$ oraz ciało rozszerzone $\mathbb{F}_{2^m}, m \in \{1, 2, 3, \ldots\}$. 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

Tablica 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

3.3 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).

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

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

3.4 Wykorzystanie w standardach Ethernetowych

Różne kody Reeda-Solomona są wykorzystywane w wielu standardach Ethernet, wyróżnione w tablicy 2

Tablica 2: Kodowania RS w różnych standardach [3]

Kodowanie RS	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

3.5 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

3.6 Tworzenie kodu

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

3.6.1 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 $\mathbb{F}_q$.

3.6.2 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.

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

3.7 Dekodowanie

3.7.1 Algorytm Berlekampa-Welcha

W roku 1986 Lloyd R. Welch oraz Elwyn R. Berlekamp uzyskali patent na dekoder umożliwiający uzyskanie oryginalnego wielomianu $p_m(x)$ oraz wielomianu $E(x)$ który zwraca 0 dla punktów x w których nastąpiło przekłamanie [4]

4 Modulacja

4.1 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 FSK, 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.

4.2 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.

4.3 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 [5].

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.

4.4 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 przepustowość przy niezmienionej szerokości pasma. Poszczególne techniki PAM różnią się między sobą liczbą wykorzystywanych poziomów modulacji. 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.

4.4.1 PAM3

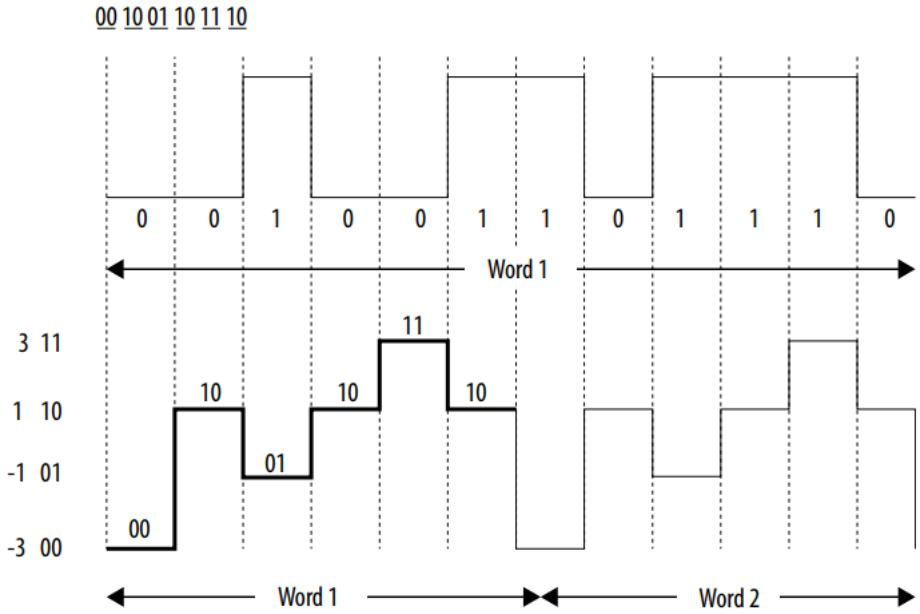
PAM3 — trzypoziomowa modulacja PAM — do przesyłu wykorzystuje wartości +1, 0, −1. Jeden symbol koduje $log_2 3 \approx 1,58$ bit/symbol. PAM3 użyty został m.in. w standardach 100BASE-T4 (wczesna implementacja Fast Ethernet) i BroadR-Reach Ethernet standard — wykorzystywany w branży automotive, opracowany przez firmę Broadcom Corporation. Nie jest to schemat modulacji PAM, który często pojawia się w praktycznych rozwiązaniach, z uwagi na to, że technologie wykorzystujące tą 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 $\frac{11/7}{log_2 3} * 100\% \approx 99\%$

4.4.2 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 [6]:



Warto zwrócić uwagę, 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 wykorzystywany jest m.in. w technologiach 100, 200 i 400 Gigabit Ethernet.

4.4.3 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. Wykorzystywana jest w technologiach 10GBASE-T, 25GBASE-T czy 40GBASE-T.

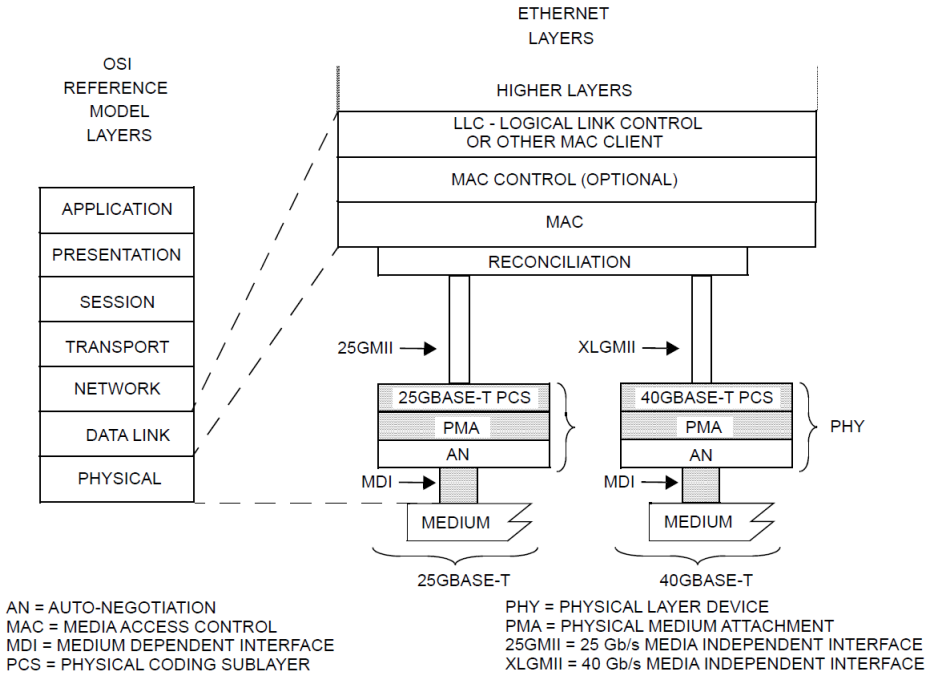
5 40GBASE-T

5.1 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.

5.2 Położenie 40GBASE-T w modelu OSI



5.3 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.

5.4 Warstwa PCS

40GBASE-T PCS (Physical Coding Sublayer) jest warstwą odpowiedzialną m.in. za kodowanie i dekodowanie oraz skramblowanie i deskramblowanie. W jej skład wchodzą m.in. skrambler i deskrambler, kodery i dekodery RS-FEC i LDPC oraz układ mapujący bity na 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 (Low density parity check). 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).

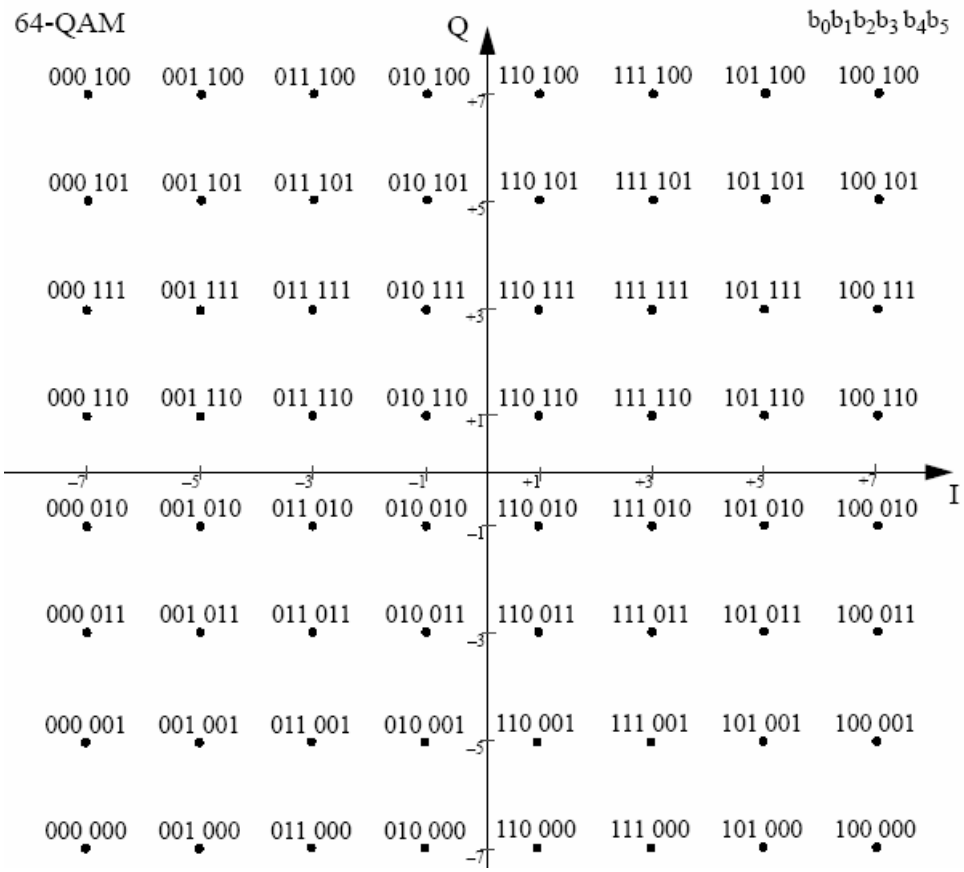
5.5 Modulacja w 40GBASE-T

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.

5.5.1 Double Square Quadrature Amplitude Modulation

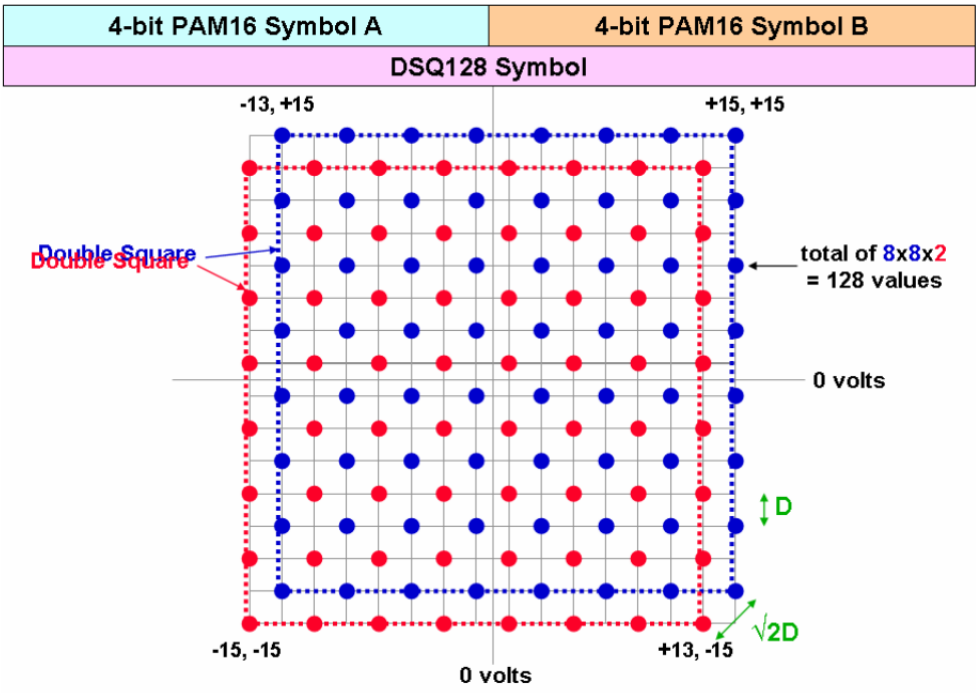
Jak opisano w 5.4, warstwa PCS koduje otrzymane dane w 7-bitowe grupy. W technologii 40GBASE-T symbole nadawane przez warstwę PMA wybierane są z konstelacji DSQ128 (Double Square Quadrature Amplitude Modulation).

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:



Diagam 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. 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 5.5.2



5.5.2 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:

$$\mathrm{PAM16}_1 = 2y_1 - 15$$

$$\mathrm{PAM16}_2 = 2y_2 - 15$$

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

Pair A	$\mathrm{PAM16}_1$<0>	$\mathrm{PAM16}_2$<0>	$\mathrm{PAM16}_1$<4>	$\mathrm{PAM16}_2$<4>	...	$\mathrm{PAM16}_1$<508>	$\mathrm{PAM16}_2$<508>
Pair B	$\mathrm{PAM16}_1$<1>	$\mathrm{PAM16}_2$<1>	$\mathrm{PAM16}_1$<5>	$\mathrm{PAM16}_2$<5>	...	$\mathrm{PAM16}_1$<509>	$\mathrm{PAM16}_2$<509>
Pair C	$\mathrm{PAM16}_1$<2>	$\mathrm{PAM16}_2$<2>	$\mathrm{PAM16}_1$<6>	$\mathrm{PAM16}_2$<6>	...	$\mathrm{PAM16}_1$<510>	$\mathrm{PAM16}_2$<510>
Pair D	$\mathrm{PAM16}_1$<3>	$\mathrm{PAM16}_2$<3>	$\mathrm{PAM16}_1$<7>	$\mathrm{PAM16}_2$<7>	...	$\mathrm{PAM16}_1$<511>	$\mathrm{PAM16}_2$<511>

6 Zajęcia dydaktyczne

6.1 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.
   - Odpowiedź: Tablica 1 bądź (11) oraz (12)
2. Czym się różni słowo kodowe wygenerowane kodem systematycznym i niesystematycznym
   - słowo kodowe w kodowaniu systematycznym w przeciwieństwie do niesystematycznego zawiera w sobie kodowaną wiadomość (3.6.2)
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
   - Zaleta: jeden symbol koduje więcej bitów - więcej bitów jest przesyłanych w jednym cyklu zegara
   - Wada: różnica między poszczególnymi symbolami maleje, a więc trudniej jest rozróżnić symbole
5. Czym się różni szerokość pasma od przepustowości?
6. Opisz krótko czym jest NRZ (Non-Return-to-Zero)
7. Czym jest przepływność łącza?
8. Która biblioteka wykorzystywana jest do stworzenia GUI?
   A: Tkinter
   B: PyQt — Odpowiedź prawidłowa w punkcie 2.1
   C: OpenGL
   D: WindowsForms

6.2 Narzędzia

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

6.3 Ćwiczenie dydaktyczne — modulacje PAM

6.3.1 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.

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 działa 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 "gmatwaczy"bitów, które zamienią takie sekwencje na bardziej zróżnicowany ciąg np. skrambler.

6.3.2 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.

6.3.3 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 (fffffff ...). Popatrz na wynik symulacji. Jak nazywa się zaobserwowane zjawisko? Czy znasz sposoby, które zapobiegają jego wystąpieniu? Zanotuj w sprawozdaniu.

6.4 Ćwiczenie dydaktyczne — Kodowanie Reeda-Solomona

6.4.1 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 $\mathrm{GF}(q)$.

6.4.1.1 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.

6.4.1.2 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	+	·
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

6.4.1.3 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:

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

6.4.1.4 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"

6.4.1.5 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.

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

6.4.1.6 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 $g(x)$

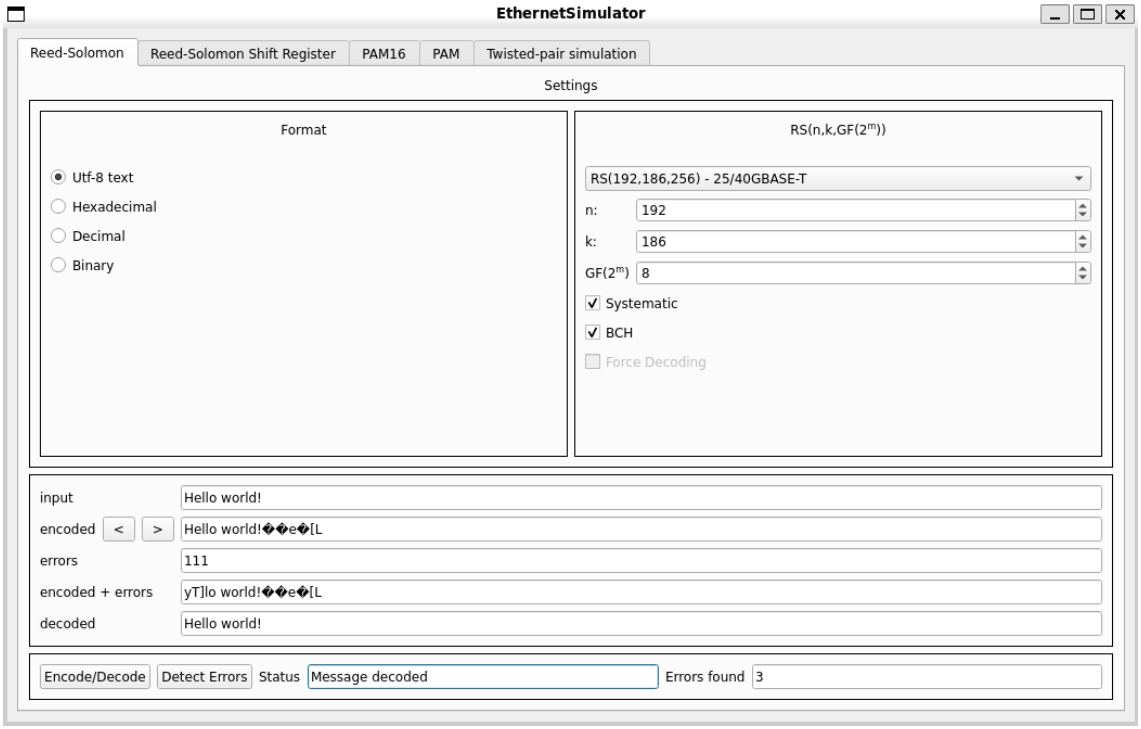
6.4.1.7 systematyczny kod BCH

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

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

6.4.2 Narzędzia

6.4.2.1 Reed-Solomon



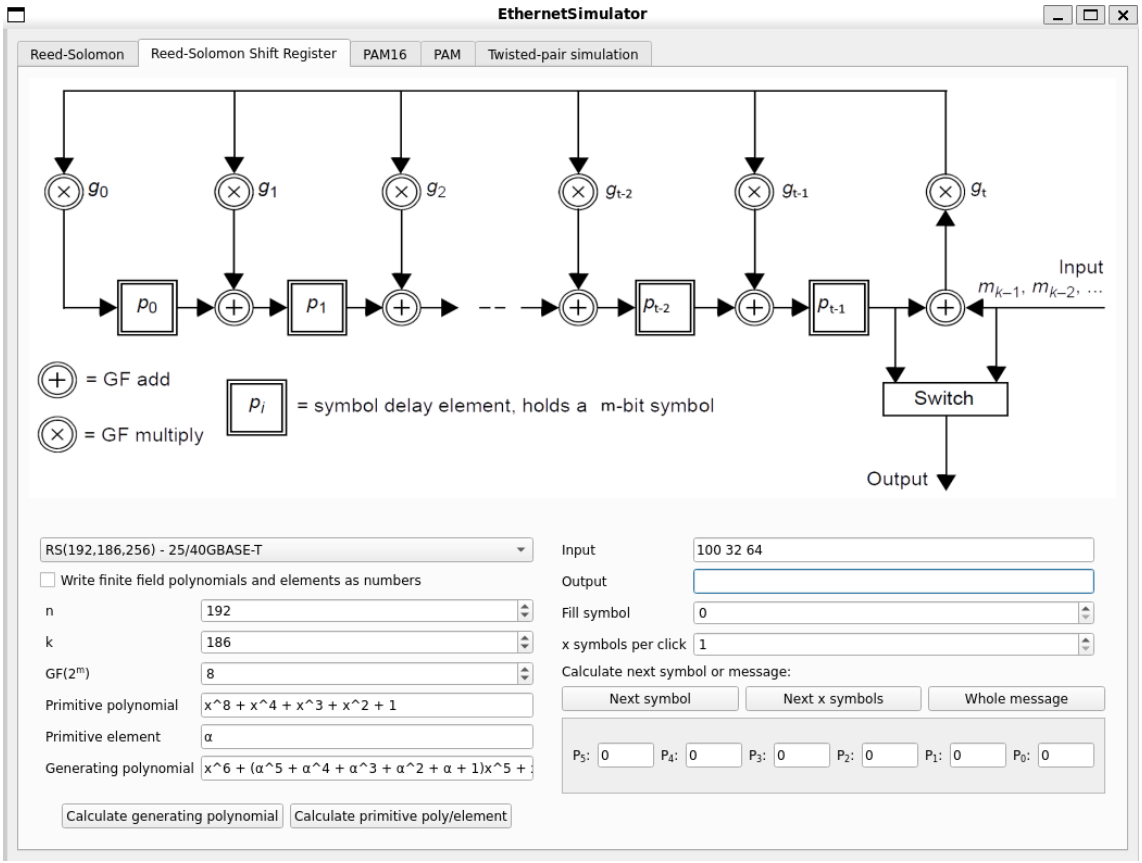
Zakładka Reed-Solomon składa się z 3 części

- 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.

6.4.2.2 Reed-Solomon Shift Register



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.

6.4.3 Zadania

- Zakładka: Reed-Solomon, ustawienia programu: kod systematyczny BCH, $n = 7$, $k = 2$, $GF = 2^3$. Oblicz słowo kodowe dla wiadomości składającej się z liczb dziesiętnych ‘3 1’. Zwiększ k o 1 i dodaj k-ty symbol ze słowa kodowego do wiadomości. Powtarzaj aż do $k = 6$. Zapisz słowa kodowe i zanotuj spostrzeżenia. Czy wiesz dlaczego otrzymałeś takie słowa kodowe?

- 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: 25/40GBASE-T, kod systematyczny BCH. Wpisz wiadomość do zakodowania: 'hello world' i zmień format na szesnastkowy. Sprawdź i zapisz ile błędnych symboli koder jest w stanie poprawić i wykryć

- Zakładka: Reed-Solomon Shift Register, ustawienia programu: $n = 7$, $k = 3$, $GF = 2^3$. Oblicz wielomiany prymitywne automatycznie naciskając przycisk 'Calculate primitive poly/element'. Zakoduj i zapisz 3 dowolne niezerowe wiadomości. Przejdź do zakładki Reed-Solomon i powtórz działania używając tych samych wiadomości oraz używając kodera systematycznego BCH z tymi samymi n, k, GF.

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] Lloyd R. Welch and Elwyn R. Berlekamp. Error correction for algebraic block codes, U.S. Patent 4 633 470, Dec. 30, 1986.

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

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