Politechnika Wrocławska

Wydział Elektroniki

Niezawodność i Diagnostyka Układów Cyfrowych 2 - Projekt

System ARQ

Autor:

Jakub Tołściuk, 248845

Prowadzący:

dr hab. inż. Henryk Maciejewski

1. Cel i założenia projektu

a) Model kanału transmisyjnego

- Model BSC (Binary symmetric channel) (błędy pojedyncze) - kanał ten będzie przyjmował bity na wejściu i z pewnym prawdopodobieństwem zmieniał konkretne bity na przeciwne. Następnie bity te będą przekazywane na wyjście.

Funkcja ta przyjmuje dwa argumenty. Ciąg zer i jedynek jako pakiet oraz prawdopodobieństwo przekłamania bitu.

- Model Gilberta (błędy grupowe) - kanał ten będzie przyjmował bity na wejściu i z pewnym prawdopodobieństwem zmieniał podciąg (o konkretnej długości) bitów. W tym podciągu każdy bit zostanie zamieniony na przeciwny.

Argumenty funkcji: „n" - pakiet, „m" - długość podciągu, „p" - prawdopodobieństwo wystąpienia takiego podciągu "przekłamań" (szum)

b) Kody detekcyjne badane w projekcie

- bit parzystości

- kod Golay'a

- kod BCH

c) Środowisko programistyczne i biblioteki

- Python 3 (w środowisku programistycznym PyChan)

- biblioteka "NumPy" (Wykorzystana została np. do operacji na tablicach)

- biblioteka „komm" (Wykorzystana została do kodowania i dekodowania pakietów)

- biblioteka „time" (do pomiaru czasu wykonania eksperymentu)

- biblioteka „xlsxwriter" (do automatycznego zapisywania wyników eksperymentu w pliku .xlsx)

d) Dane na wyjściu programu (zapisywane w pliku .xlsx)

- BER (Bit error rate) – stosunek bitów „przekłamanych" na wyjściu sygnału do wszystkich wysłanych bitów

- Nadmiar kodowy – raportowany jako wartość „A" (jest to stosunek liczby wszystkich przesłanych bitów i liczby bitów wiadomości do przesłania)

- Szybkość symulacji (w sekundach)

- Informacja o tym jakie było prawdopodobieństwo przekłamania bitu w danym pomiarze

2. Opis wykonanego symulatora

Symulator składa się z trzech głównych części:

-> Implementacja bazowych metod potrzebnych do przeprowadzenia symulacji (np. porównanie bitów wejściowych z bitami wyjściowymi, metody związane z kodowaniem/dekodowaniem) [2.1]

-> Implementacja metod, które wykonują pojedyncze symulacje (w zależności od użytego kodowania oraz kanału transmisyjnego) [2.2]

-> Implementacja metod, które wykonują eksperymenty, czyli określoną liczbę symulacji [2.3]

2.1 Podstawowe metody potrzebne do symulacji

-> **Wygenerowanie wiadomości** [Metoda: „generateMessage(messageLength, packetLength)”]

Metoda ta generuje wiadomość o podanej liczbie bitów (messageLength). Wiadomość jest dzielona na pakiety o podanej długości (packetLength) [liczonej w bitach]. Liczba pakietów jest liczona jako wynik dzielenia długości wiadomości przez długość pakietu (z zaokrągleniem w górę).

Została tutaj użyta biblioteka numpy do podziału wiadomości na pakiety oraz losowania wartości bitu (0 lub 1).

-> **Kanały transmisyjne**

a) BSC

Kanał ten przyjmuje (jako argumenty) pakiet oraz prawdopodobieństwo przekłamania bitu. Do implementacji tego kanału została użyta biblioteka „komm”.

b) Burst Error Channel

W tym kanale dodatkowo potrzebny jest argument, który określa długość podciągu, który ma zostać przekłamany. Długość tego podciągu zostaje uwzględniona przy analizie wyników.

-> Metody do kodowania i dekodowania

Za pomocą metod „addParityBit” i „checkParityBit” można kodować i dekodować pakiety z użyciem bitu parzystości. Zostały one zaimplementowane z użyciem biblioteki „numpy”. Natomiast pozostałe metody w tej części (takie jak np. „generateCRCGolay()”) zwracają jedynie obiekt, który został stworzony za pomocą biblioteki „komm” (klasa „CyclicCode”). Obiekt ten umożliwia zarówno kodowanie jak i dekodowanie pakietu (w kontekście konkretnego generatora wielomianu).

-> Porównanie wiadomości [Metoda: compareData(data1, data2)]

Jest to ostatnia metoda w tej część kodu. Służy ona do porównania dwóch wiadomości w celu zliczenia bitów o różnych wartościach. W późniejszej części kodu, na podstawie tej informacji, liczony jest „Bit error rate”.

2.2 Metody wykonujące symulacje

Ta część kodu składa się z sześciu podobnych metod. Jest ich tyle ponieważ użyte zostały trzy sposoby kodowania i dwa kanały transmisyjne a każde kodowanie jest testowane w zakresie każdego kanału. Wejścia i wyjścia tych metod zostaną omówione w punkcie „3”.

2.3 Metody przeprowadzające serie symulacji

W tej części ponownie zostało zaimplementowane sześć metod (po jednym eksperymencie dla każdej symulacji). Określona jest tutaj liczba pomiarów oraz wszystkie parametry, które dotyczącą poszczególnych symulacji. Dodatkowo zastosowana została metoda „writeInExcel", która ułatwia zapisywanie wyników eksperymentu w excelu. W tej części programu została wykorzystana biblioteka „xlsxwriter".

3. Organizacja eksperymentu symulacyjnego

Organizację eksperymentu symulacyjnego omówię na przykładzie kodowania bitem parzystości dla kanału „BSC" oraz „Burst error".

a) W kontekście kanału „BSC"

Metoda „simulationBSCandParityBit(...)" przyjmuje cztery argumenty. Długość wiadomości i długość pakietu (do wygenerowania), maksymalna liczba retransmisji oraz prawdopodobieństwo przekłamania bitu w użytym kanale (BSC).

Zwracane są natomiast cztery wartości:

-> Bit error rate - obliczany jako liczba przekłamanych bitów (z użyciem metody „compareData(…)") podzielona przez długość wiadomości. Następnie wartość ta jest mnożona razy 100 aby uzyskać wynik w procentach.

-> Wartość „A" - określona ona nadmiar kodowy. Jest to stosunek liczby wszystkich przesłanych bitów (z uwzględnieniem retransmisji) i liczby bitów wiadomości do przesłania.

-> Czas symulacji - czas (liczony w sekundach), który był potrzebny na wykonanie wszystkich działań związanych z kodowanie, przetworzeniem przez kanał transmisyjny oraz dekodowaniem wiadomości.

-> Prawdopodobieństwo przekłamania bitu - jest pomnożone na końcu razy 100, żeby było wyrażone w procentach.

b) W kontekście kanału „Burst error"

Ta symulacji różni się tylko dodaniem kolejnego parametru na wejściu, który określa długość podciągu przekłamanych bitów. Jest to następnie użyte jako argument dla kanału transmisyjnego.

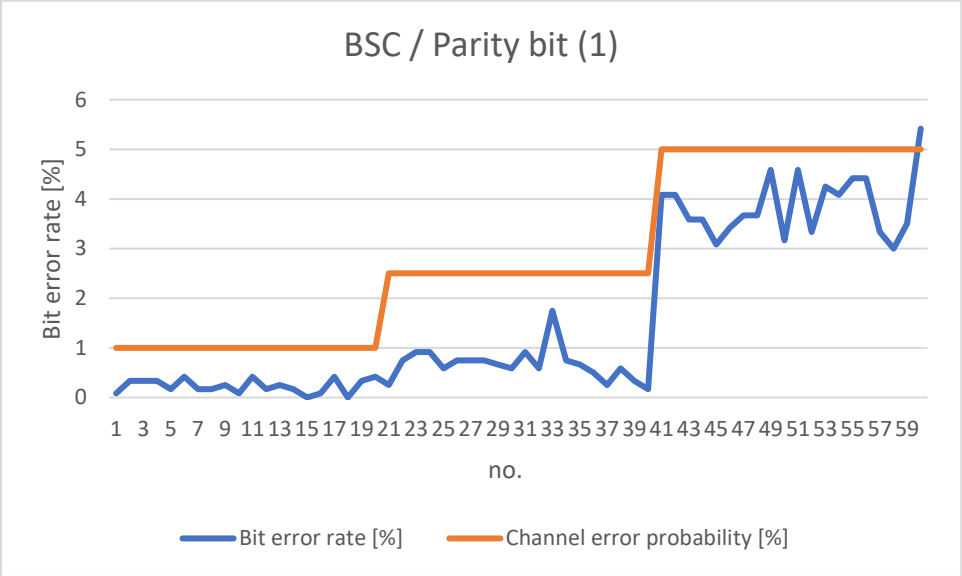
4. Wyniki oraz ich analiza

[Uwaga: W przeprowadzonych pomiarach założyłem, że długość pakietu i wiadomości jest stała (w kontekście użytego sposobu kodowania)]

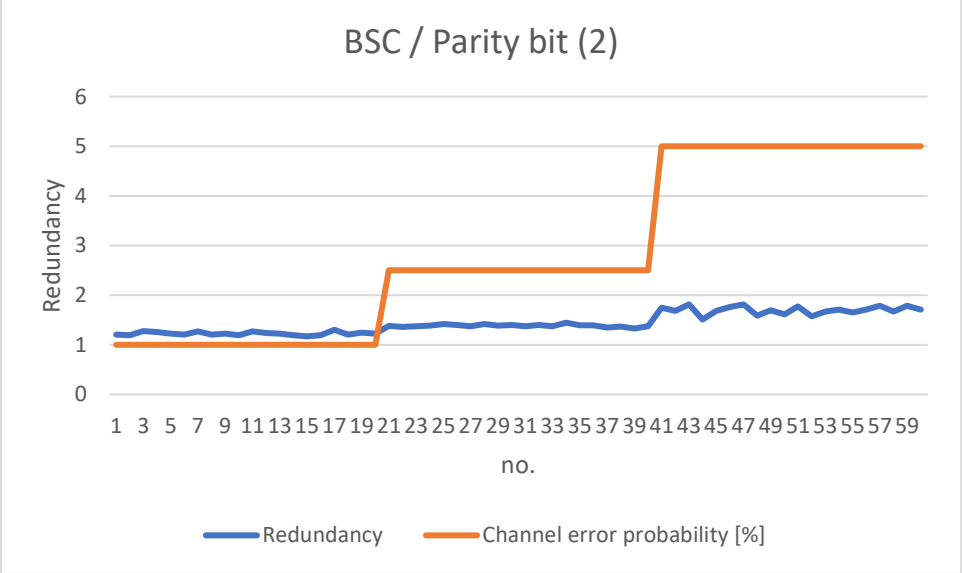
4.1 Kodowanie za pomocą bitu parzystości

a) Kanał błędów pojedynczych (BSC)

-> Bit error rate

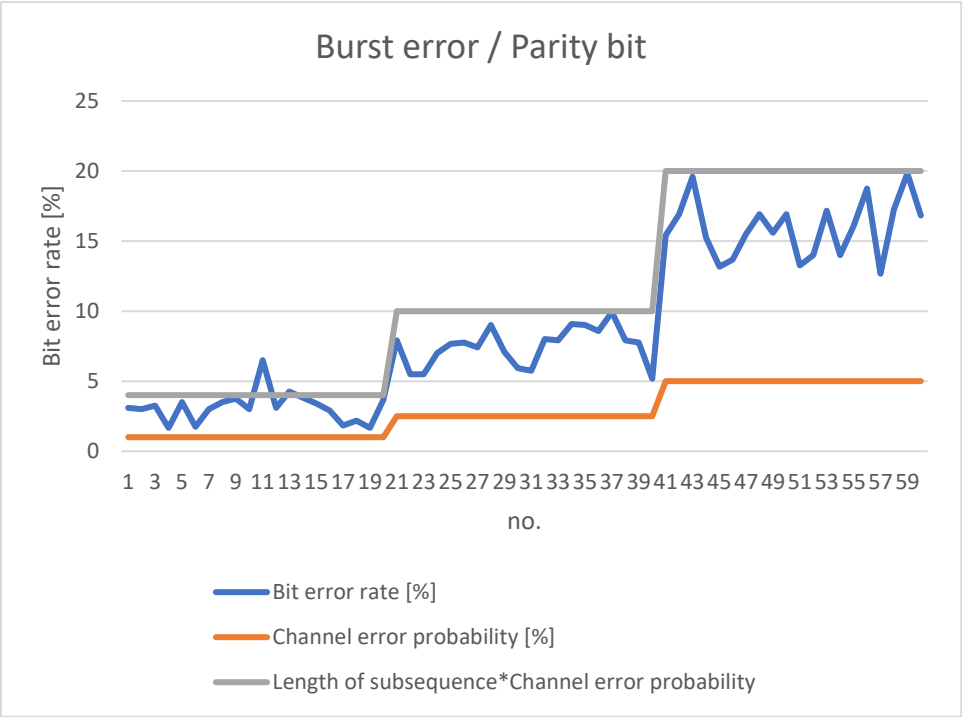


-> Nadmiarowość

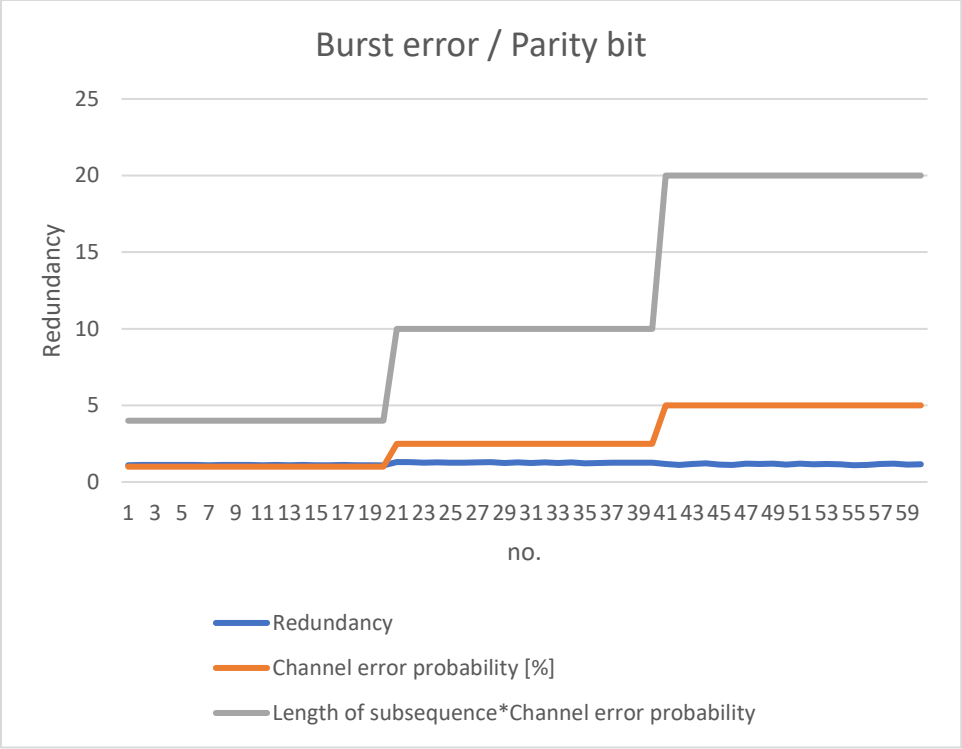


b) Kanał błędów grupowych (Burs error)

-> Bit error rate



-> Nadmiarowość



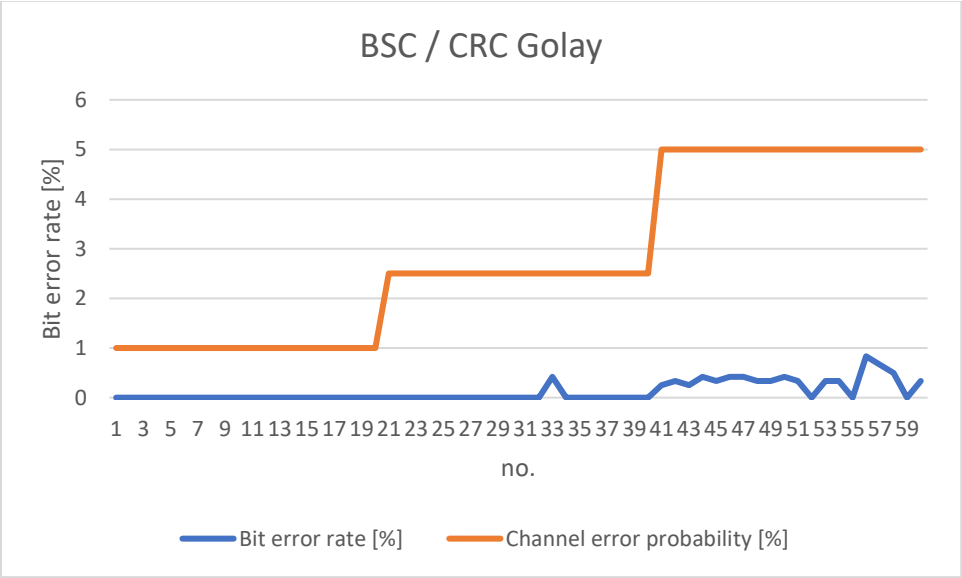
Największą zaletą kodowania za pomocą bitu parzystości jest mała redundancja. Warto jednak zauważyć, że redundancja dla kanału błędów grupowych jest dużo mniejsza niż dla kanału błędów pojedynczych. W tym kodowaniu celowo dobrałem długość przekłamanego

podciągu o parzystej wartości (4) aby pokazać istotną wadę tego kodowania. W przypadku parzystej liczby przekłamanych bitów nie zostanie wykryty błąd w pakiecie ponieważ parzystość liczby bitów o wartości „1” się nie zmieni (zatem nie zostanie wykonana retransmisja). Jednak czasami dochodzi do sytuacji, w której np. ostatnie trzy bity pakietu (z czterech) zostały przekłamane, więc błąd zostanie rozpoznany (takie przypadki zdarzają się rzadko).

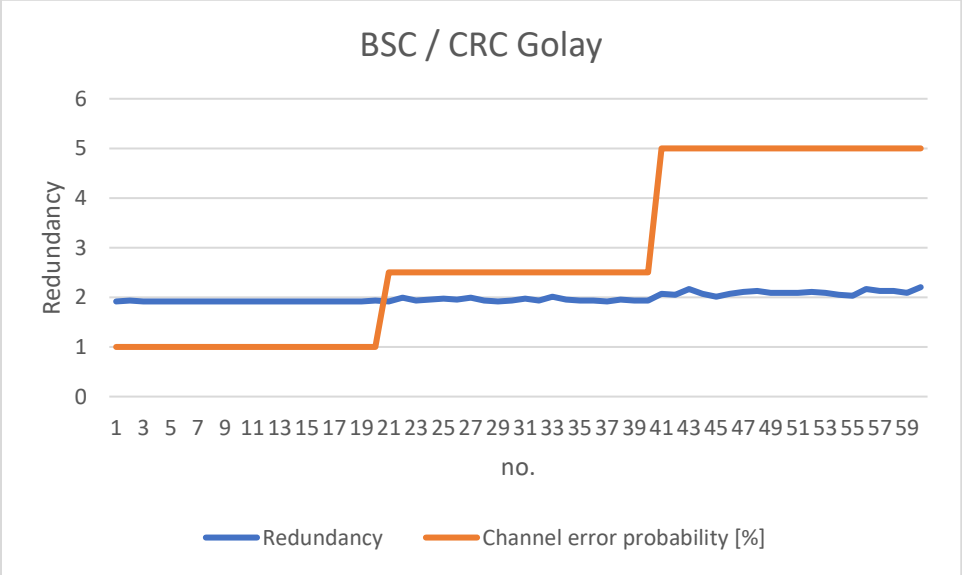
4.2 Kodowanie za pomocą CRC Golay

a) Kanał błędów pojedynczych (BSC)

-> Bit error rate

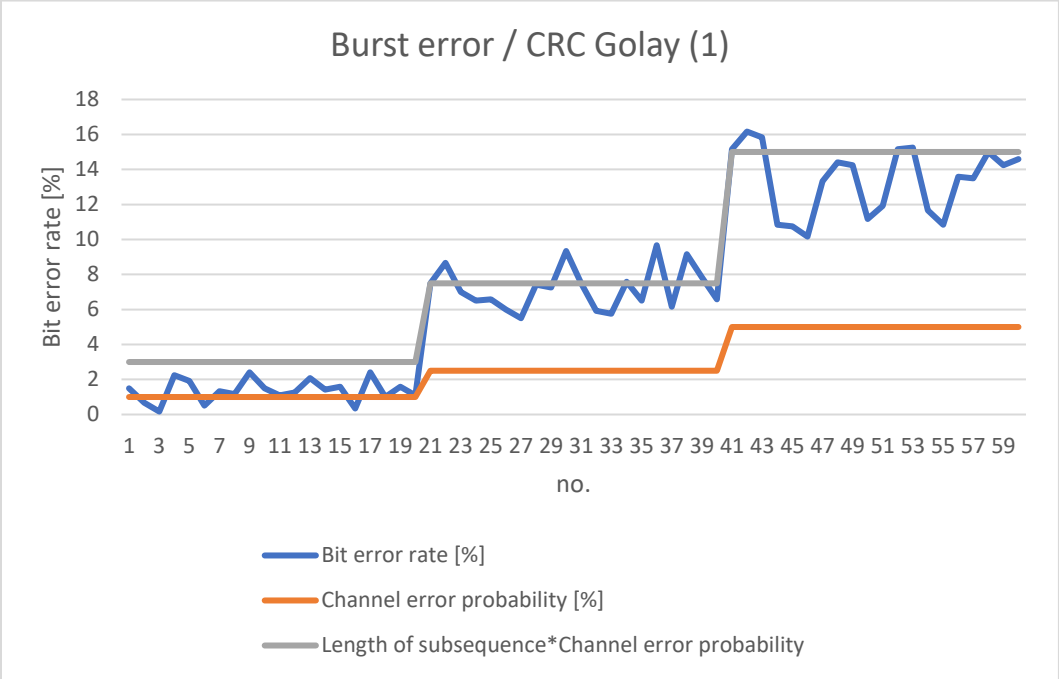


-> Nadmiarowość

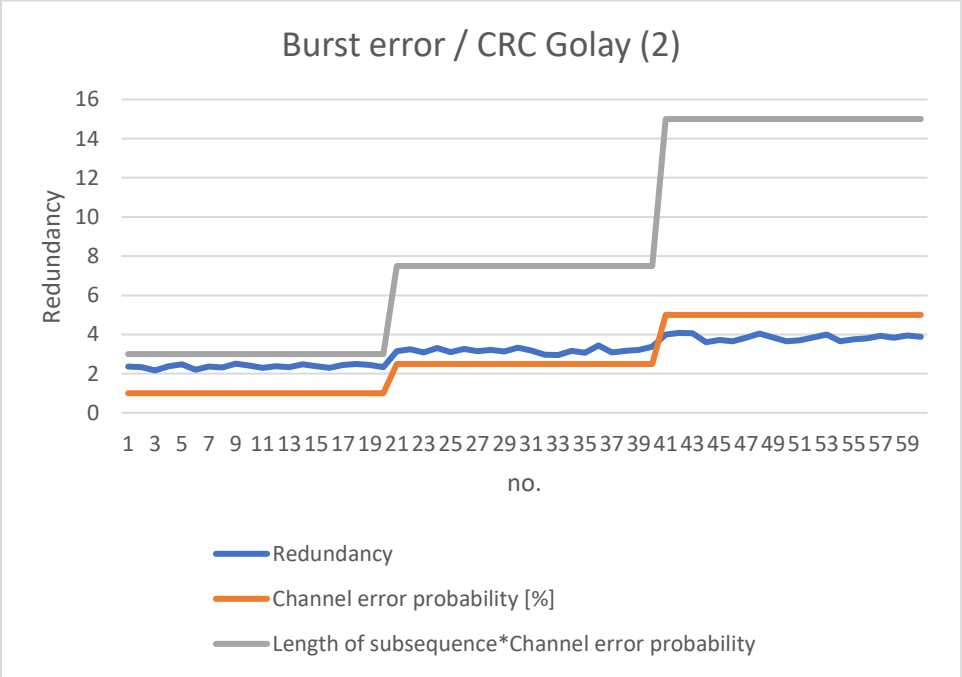


b) Kanał błędów grupowych (Burst error)

-> Bit error rate



-> Nadmiarowość

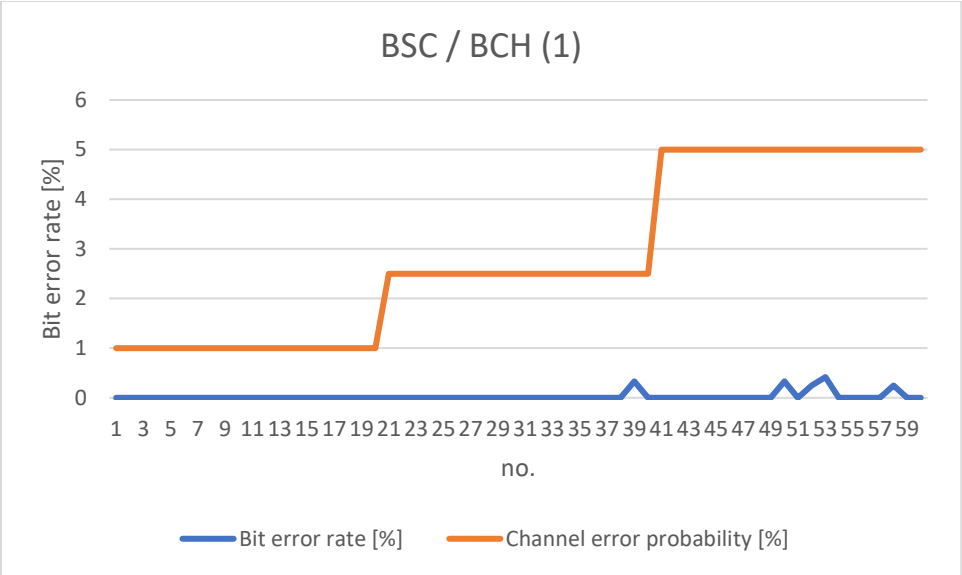


Nadmiarowość dla kodowania za pomocą CRC Golay jest dużo większa niż dla bitu parzystości ale w przypadku kanału BSC liczba popełnionych błędów jest znacząca mniejsza. Natomiast liczba błędów w kontekście kanału błędów grupowych jest porównywalnie duża jak w przypadku bitu parzystości (pomimo większej redundancji).

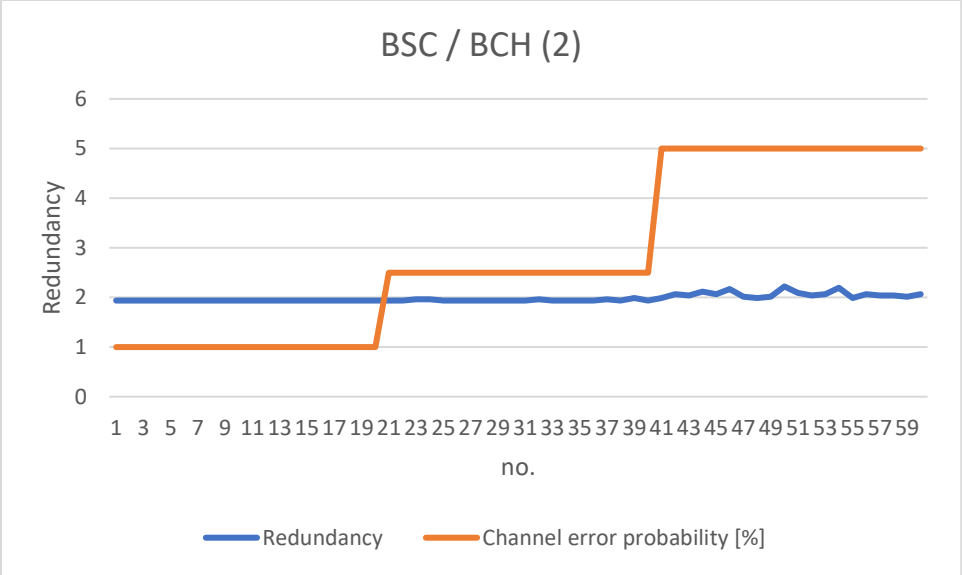
4.3 Kodowanie za pomocą kodu BCH

a) Kanał błędów pojedynczych (BSC)

-> Bit error rate

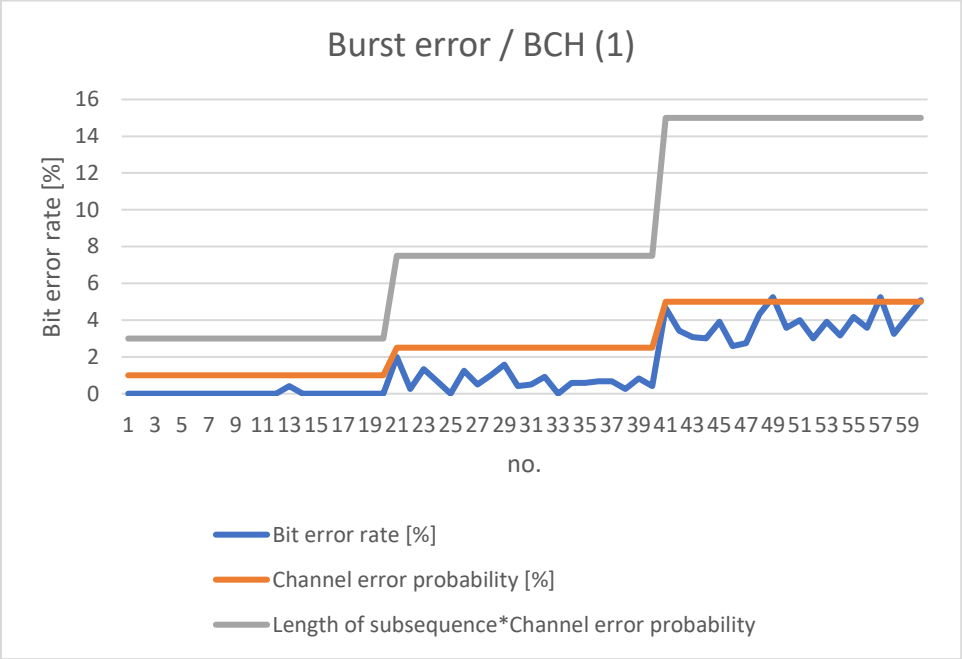


-> Nadmiarowość

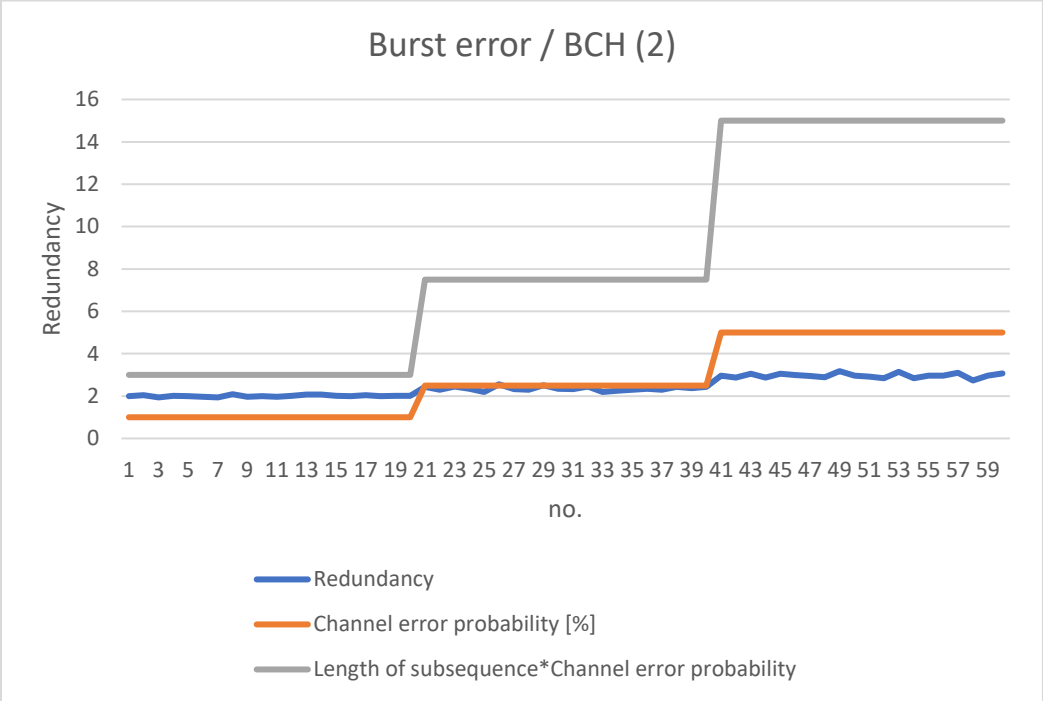


b) Kanał błędów grupowych (Burst error)

-> Bit error rate



-> Nadmiarowość



Nadmiarowość kodowania za pomocą BCH jest dużo większa niż w przypadku kodowania za pomocą bitu parzystości ale zarówno dla kanału BSC jak i Burst error otrzymane wyniki „Bit error rate" są bardzo dobre. Ze wszystkich trzech sposobów kodowania w tym została popełniona najmniejsza liczba błędów.

5. Wnioski i uwagi

Na podstawie przeprowadzonych eksperymentów stwierdzam, że w przypadku kanału transmisyjnego, w którym występują małe zakłócenia, tzn. prawdopodobieństwo przekłamania bitu jest małe, warto użyć kodowania za pomocą bitu parzystości. Ma to na celu zmniejszenie nadmiarowości a co za tym idzie zwiększenie szybkości transmisji.

Natomiast w przypadku gdy prawdopodobieństwo przekłamania bitu jest wysokie należałoby użyć kodu BCH. Osiągnął on zdecydowanie najlepsze wyniki w kontekście „Bit error rate” (kosztem redundancji).