Jakub Sobieszek 232838
Adrian Śnieżek
prow. Piotr Semberecki

04.06.2019
Wrocław

# Niezawodność i Diagnostyka Układów Cyfrowych 2

# Projekt

1.Cele Projektu

Tematem Projektu była symulacja i implementacja systemu ARQ – Automatic Repeat Request Zadaniem Aplikacji było wczytanie zdjęcia z pliku, podzielenie go na paczki bitów, następnie symulowanie przesyłania paczek do odbiorcy poprzez funkcje wprowadzające błędy na podstawie różnych modeli błędów. Przesyłanie odbywa się poprzez jeden z systemów detekcji i korekcji błędów, który na podstawie sumy kontrolnej decyduje czy przyjąć paczkę, czy zażądać jej ponownego przesłania. Ostatecznie, przyjęte paczki scalane były z powrotem do formy obrazu. W zależności od wybranych modeli błędów, protokołów detekcji i korekcji, sumy kontrolnej, oraz innych parametrów programu można było obserwować stopień zniekształcenia obrazu wyjściowego. W programie zastosowano:

Modele błędów:

- -Binary Symmetric Channel
- -Gilbert's model

Protokoły detekcji i korekcji:

- -Stop and Wait
- -Go back n
- -Selective Repeat

Sumy kontrolne:

- -parity bit
- -crc32

dodatkowymi parametrami programu były:

- -prawdopodobieństwo wystąpienia błedu
- -ilość możliwych ponownych przesłań, po wyczerpaniu których odbiorca zapisywał sztywno ustaloną paczkę (np. paczkę składającą się z samych zer)
- -wielkość ramki wg której przesyłane będą ramki (dotyczy protokołów Selective Repeat oraz Go back n)

2.Realizacja

Paczka zrealizowana została jako klasa, z atrybutami (m.in.):

- -macierzy jednowymiarowej zawierającej w sobie ciąg bitów (dane obrazu)
- -sumy kontrolnej

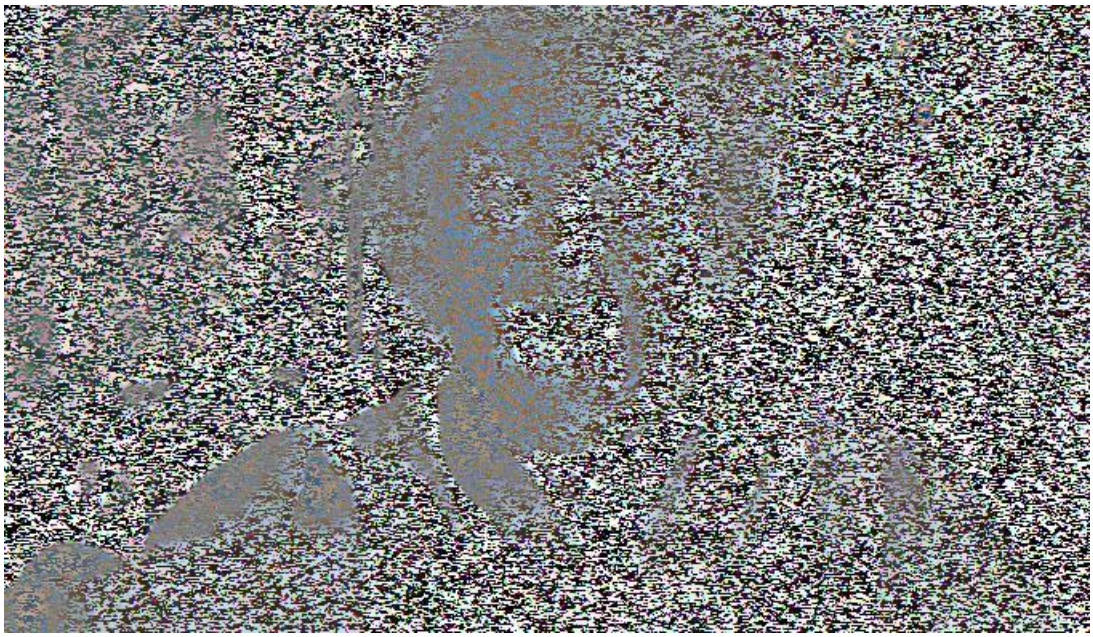
oprócz w/w używane były atrybuty sytuacyjne potrzebne w zależności od używanego protokołu.

Po uruchomieniu programu użytkownik jest proszony o wybór parametrów programu: prawdopodobieństwa błędu, sumy kontrolnej, modelu błędów etc. Następnie zaczytane zdjęcie zostaje przesłane wedle wybranych parametrów. Po zakończeniu transmisji generowany jest log operacji ze statystykami, oraz zapisany oraz wyświetlony zostaje obraz wyjściowy.

3.Wyniki

Parametry:
prawdopodobieństwo błędu: 5.0%
ilość możliwych ponownych przesłań: 5
Protokół Stop and Wait
suma kontrolna crc32

Pierwszy obraz przesłany z modelem błędów Gilberta, drugi modelem symetrycznym (BSC)
Widać wyraźnie kumulowanie się błędów zgodnie z założeniami modelu Gilberta (obraz 1) co powoduje większy stopień zakłócenia obrazu, oraz grupowanie się błędów.



Parametry:
Prawdopodobieństwo błędu 2.0%
Ilość możliwych ponownych przesłań - 5
protokół Selective Repeat
ramka rozmiaru 5 paczek
model błędów Gilberta

Pierwszy obraz przesłany z sprawdzaniem poprawności za pomocą bitu parzystości, drugi za pomocą sumy crc32. Od razu widać ogromnie niską efektywność bitu parzystości jako testu poprawności. Wynika to głównie z faktu, iż jeśli w paczce zostanie przekłamana parzysta ilość bitów, pit parzystości będzie się zgadzał i paczka zostanie zatwierdzona jako poprawna. Niezwykle nieefektywne jest używanie bitu parzystości przy modelu błędów Gilberta, w którym po wystąpieniu pierwszego błędu, oczekiwać możemy bardzo długiego ciągu błędów. Jeśli pierwszy błąd wystąpi np. w 5 bicie paczki $x_{i,}$ to nawet jeśli ta paczka zostanie wychwycona jako zniekształcona, możemy oczekiwać że jakaś ilość paczek które zostaną wysłane po niej będzie posiadać zniekształcony każdy bit, w przypadku paczek 1 bajtowych jest to parzysta ilość.

Parametry:
brak możliwości ponownego przesłania – zerowa korekcja błędu
protokół Stop and Wait
model błędów Gilberta
suma kontrolna Crc32

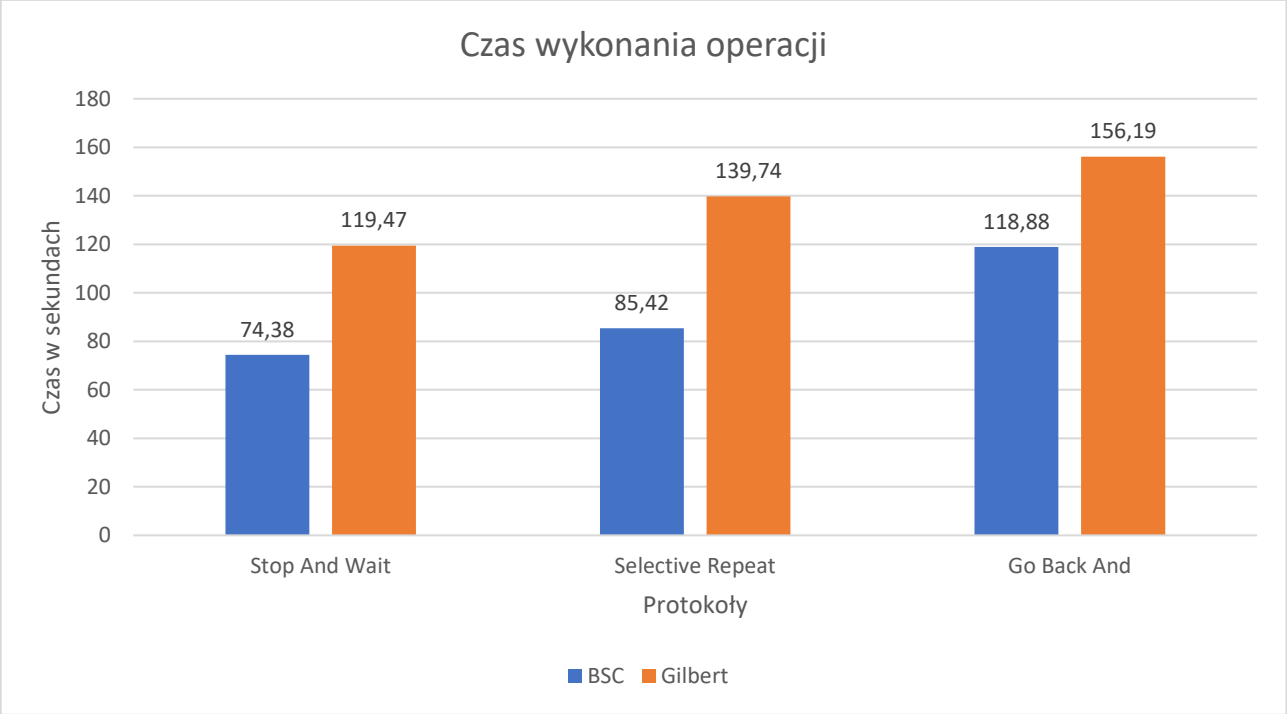
Pierwszy obraz przesłany z prawdopodobieństwem błędu 2%, drugi 0.2%. Ponieważ przesyłanie odbywa się najpierw po szerokości a następnie po wysokości obrazu, widać, że przy większym p.podobieństwie błędy pojawiają się częściej, ale też szybciej „znikają". Obserwujemy skupiska o dość nieregularnych kształtach. Przy niższym z kolei, błędy pojawiają się rzadko, ale też bardzo długo zajmuje, aby przesłana paczka nie została zniekształcona, przez to obserwujemy błędy układające się w długie poziome linie na obrazie, tworzące większe skupiska, ale występujące rzadziej. W przypadku niższego prawdopodobieństwa ogólna ilość błędów jest niższa o ok 10%

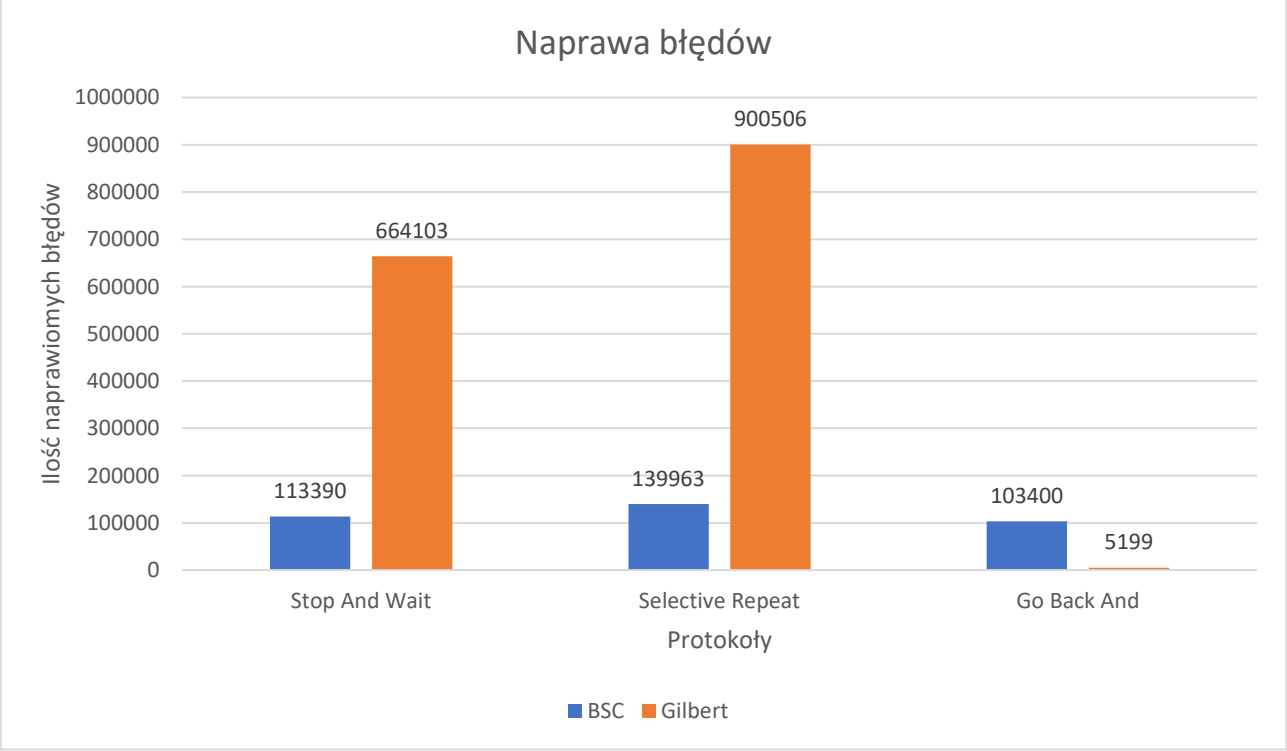
4. Czas i błędy

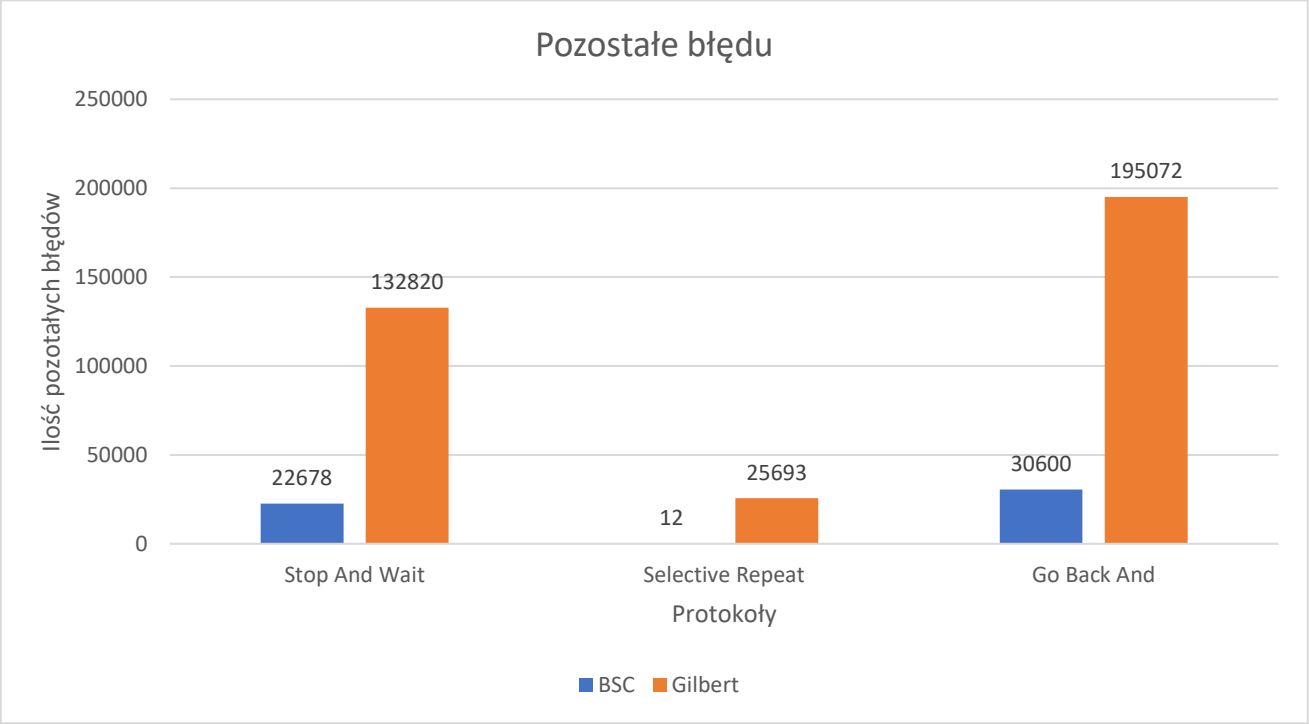
Na podstawie serii testów została wyznaczona średnia z zebranych wyników
( czasu, ilości ramek poprawionych oraz nie poprawionych ) podsumowując jakość i prędkość działania poszczególnych protokołów.

Parametry zastosowane do testów:
- prawdopodobieństwo błędu 2%
- 5 możliwych ponownych przesłań
- paczka zawierająca 10 ramek

Zestawienie wyników na wykresach






Na podstawie powyższych wykresów można stwierdzić, że najkorzystniejszym protokołem jest Selective Repeat. Osiąga on najlepsze wyniki w poprawianiu błędów mimo, że nie jest najszybszy. W kwestii prędkości wygrywa Stop and Wait, który naprawia błędy podobnie do Selective Repeat'a jeżeli porównując przypadek BSC. Patrząc na błędy powstałe przez model Gilberta, Selective Repeat radzi sobie o wiele lepiej niż pozostałe protokoły. Najmniej korzystnym protokołem okazuje się Go Back And co nie jest większym zaskoczeniem. Protokół ten wysyła ramki paczkami i jeżeli przekroczy limit możliwych ponownych przesyłów, cała paczka jest kodowana jako źle wysłana. Zmniejszenie wartości ilości ramek w jednej paczce powoduje wzrost wydajności protokołu, ale wciąż nie zbliża się do jakości i potencjału Selective Repeat.