Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. E. Grass
D. Cvetkovski (darko.cvetkovski@hu-berlin.de)

19.01.2018

Praktikum zur Vorlesung
Drahtlose Breitbandkommunikationssysteme

Praktikum 10
Empfang eines kompletten OFDM-Frames

Abgabe: Bis 25.01.2018 23:59 Uhr (per e-mail). Die Praktikumsaufgaben sollten möglichst in Gruppen von 2 bis 3 Personen bearbeitet werden, damit nur eine Abgabe erfolgen muss.
Die Simulationen haben mit Matlab Simulink 2016a zu erfolgen.
Bei ausreichender Bearbeitung der Aufgaben gibt es 1 Punkt.
Beachten Sie auch die aktuellen Hinweise auf der Vorlesungswebsite unter:
http://www.informatik.hu-berlin.de/forschung/gebiete/bbk

Aufgabe 1 (Zufällige Kanalverzögerung)
Erstellen Sie eine neue Simulationsdatei namens *Praktikum10.slx* und nutzen Sie als Grundlage das in *Praktikum 9* erstellt Simulationsmodell. Erzeugen Sie ein Subsystem namens *Frame Delay* vor dem Kanal in dem Sie eine zufällige Verzögerung des OFDM-Frames einführen. Zuerst verketten (d.h. verlängern) Sie für die Simulation die, vom Transmitter erzeugten OFDM-Frames, mit jeweils einem gleich langen Spaltenvektor aus komplexen Nullen. Durch die so entstehenden Pausen in der Übertragung wird verhindert, dass sich Frames beim zufälligen Verzögern überlappen können. Danach führen Sie eine zufällige Verzögerung mit Hilfe der Blöcke *Variable Integer Delay* und *Random Integer Generator* ein. Diese Verzögerung soll für alle Samples eines Frames identisch sein und maximal eine OFDM-Frame-Länge betragen. Führen Sie die Simulation durch und testen Sie die Synchronisation mit verschiedenen Kanalparametern.

Aufgabe 2 (Automatische Verstärkungsregelung)
Fügen Sie nach dem *BBK-Kanal* einen *AGC* Block ein. Als Ersatz für ein normalerweise analoges AGC-Element soll dieser den Eingangspegel für die nachfolgenden Blöcke konstant halten. Verwenden Sie dabei folgende Parameter:

- Step size: 0,3
- Desired output power (W): 1
- Averaging length: 16
- Maximum power gain (dB): 10

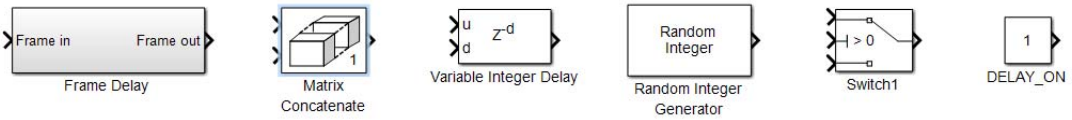
Überprüfen Sie, ob ihre grobe und feine Zeitsynchronisation auch mit dem Einschwingen des AGC zurechtkommt und modifizieren Sie diese, falls notwendig. Um die Bearbeitung der weiteren Aufgaben zu erleichtern, kann es hilfreich sein, den AGC mit einem *Switch* Block vorübergehend deaktivieren zu können. Schließen Sie danach die in Praktikum 9 erstellten *Unbuffer* und *OFDM-Synchronizer* Blöcke zwischen den Ausgang des *AGC* Blocks und den Eingang des *OFDM-Receivers* an.

Aufgabe 3 (Zeitsynchronisation)
Erstellen Sie ein Subsystem namens *Frame Parsing* innerhalb des Subsystems *OFDM-Synchronizer*, in welchem Sie synchron das eingehende Signal eines kompletten OFDM-Frames nach der Frameerkennung erfassen. Verbinden Sie das eingehende sample-basierte Signal und das *Peak Detected*-Signal als Eingänge des Subsystems. Erstellen Sie einen *Buffer* Block, der die eingehenden Samples in einem Spalten-Vektor der Größe von 1920+320=[2240] (d.h. ein OFDM-Frame plus eine Präambel) speichert. Setzen Sie den Buffer Overlap Parameter zu 2239 Samples, sodass der Ausgang des Buffers nach jedem eingehenden Sample aktualisiert wird. Verwenden Sie einen *Sample and Hold* Block, um synchron die Ausgabe des *Buffer* Blocks zu erfassen. Nutzen Sie das *Peak Detected*-Signal aus der *Fine Synchronization* als Trigger-Signal für den *Sample and Hold* Block. Dieses Signal sollte um exakt eine Länge des Datenteils eines OFDM-Frame (d.h. 1599 samples) verzögert werden, bevor es mit dem *Sample and Hold* Block verbunden wird. Extrahieren Sie das OFDM-Frame aus der Ausgabe des *Sample and Hold* Blocks, indem Sie die letzten 1920 Samples auswählen. Um die korrekte Frame-Verarbeitungsrate zu erhalten, erstellen Sie danach einen *Rate Transition* Block und setzen ihn auf *Multiple of Input Port sample time* (1920). Schließlich erstellen Sie einen *Frame Conversion* Block (Frame-Based), um mit der frame-basierten Verarbeitung des Signals fortzufahren und

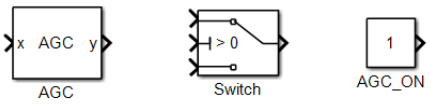
verbinden Sie den Ausgang *OFDM-Frame* mit dem *OFDM-Receiver* zur weiteren Verarbeitung mit dem zuvor entwickelten Empfänger.

Zusatzblatt zum Praktikum 10 – Nützliche Simulink Blöcke

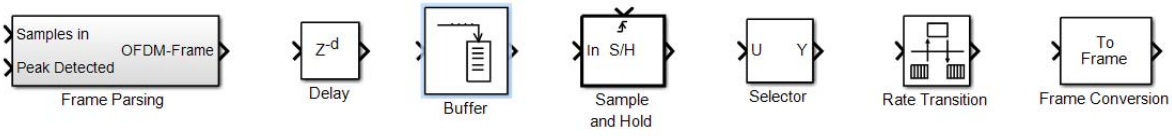
(Zufällige) Kanalverzögerung:



Automatische Verstärkungsregelung:



Zeitsynchronisation:



***Vorbereitung zum praktischen Teil des Praktikums 12 (Demonstration und Auswertung)**

- Es empfiehlt sich, dass Sie die folgenden Punkte so bald wie möglich erledigen. -

1. Einrichtung des Gitlab-Accounts

Um auf das BBK-Projekt zugreifen zu können, müssen Sie über einen aktiven Gitlab-Account verfügen. Als Informatik-Student existiert ein solcher bereits, er muss nur ggf. noch aktiviert werden. Falls Sie KEIN Informatik-Student sind, geben Sie uns bitte Bescheid, dann müssen wir das anders lösen. Nach Möglichkeit sollte ein einzelner Repräsentant für die Gruppe gewählt werden, der Zugriff auf das Gitlab-Projekt erhält. Idealerweise ist diese Person insb. bei den Praktikumsterminen anwesend.

Unter https://gitlab.informatik.hu-berlin.de/users/sign_in können Sie sich mit Ihren normalen Informatik-Credentials anmelden, die u.a. für das WLAN-Netz verwendet werden (das sind NICHT die CMS-Credentials, die man als HU-Student hat, außer, diese unterscheiden sich nicht). Stellen Sie sicher, dass "LDAP" als Anmeldeoption ausgewählt ist und melden Sie sich mit Benutzernamenkürzel und Passwort an.

Damit ist die Aktivierung des Accounts abgeschlossen und wir können Sie zu den Projekten hinzufügen. Sie erhalten ggf. eine oder mehrere Mails, wenn das geschehen ist.

2. Einrichten Ihres Projekts

WICHTIGER HINWEIS: Wir benutzen zur Koordination der Dateien das CI/CD-Environment von Gitlab, was für automatisierte Tests vorgesehen ist. Sie können sich die Umgebung ansehen, aber starten Sie bitte keine Pipelines / Jobs durch Klick auf die Play-Symbole. Da diese Jobs einerseits auf einem separaten Rechner ausgeführt werden und andererseits alle die selben Geräteadressen ansteuern, kann das zu unvorhersehbarem Verhalten führen, wenn z.B. mehrere Jobs parallel ausgelöst werden. Wir haben leider (aktuell) keine Möglichkeit, den Zugriff zu beschränken, deshalb dieser Hinweis. Ihr Teil der Interaktion mit Gitlab wird sich darauf beschränken, Binfiles dem Projekt hinzuzufügen, zu committen und die empfangenen Dateien zu erhalten. Alles andere werden wir, jedoch in Ihrem Beisein, durchführen.

Sobald wir Ihre Projekte erstellt und Sie hinzugefügt haben, können Sie über die Kommandozeile darauf zugreifen. Sie sollten das Projekt auf den Rechner laden, auf dem Sie auch Matlab/Simulink haben, um einfach die Binfiles einfügen zu können. Im Normalfall sollten das also die Rechner im Linux-Pool sein, außer, Sie haben eine eigene Lizenz auf Ihren Laptops.

Mit "git clone https://gitlab.informatik.hu-berlin.de/bbk_txrx/PROJEKTNAME.git" können Sie Ihr Projekt herunterladen, indem Sie Ihre Informatik-Credentials eingeben. Der Projektordner beinhaltet eine .sh-Datei, die Ihnen das Committen neuer Dateien vereinfachen soll, sowie eine versteckte .yml-Datei, die die Konfiguration steuert. Auch hier gilt: Sie können sich die Konfigurationsdatei ansehen, sollten diese aber im eigenen Interesse unverändert lassen.

3. Arbeiten mit dem Projekt

- Erzeugen Sie das Binfile mit Matlab (stellen Sie sicher, dass der Dateiname auf .bin endet)
- Kopieren Sie es in den Projektordner (zur selben Zeit sollte immer nur eine Datei im Projektordner liegen)
- Führen Sie das Helper-Skript mit "./githelper.sh" aus (ggf. vorher mit "chmod +x githelper.sh" ausführbar machen)

Bei erfolgreichem Commit sehen wir dann eine neue Pipeline, in der wir manuell die Übertragung starten werden.