FPGA DVB-S Encoder

Die Idee ist, einen DVB-S-Encoder in VHDL zu realisieren.

Links / Referenzen

- ETSI-Standard DVB-S
- drmpeg gr-dvbs
- TS-Erzeugung CBR

Schnittstellen

- Schnittstelle zum PC: Ethernet (UDP)
- Schnittstelle zum I/Q-Modulator: 2xDAC

Komponenten

Die geplante Komponentenstruktur wurde in KiCAD erstellt, was bei der Planung ungemein hilft:

- Download Blockschaltbild
- KiCAD-Projekt und Schaltplanfiles

Designfragen:

- Können die FrameSync-Eingänge einfach durch den Reset ersetzt werden / sind sie notwendig?

Bis nach dem Interleaver ist die Struktur byteweise, danach arbeitet sie bit-seriell. Die Pipeline muss vor dem RS-Encoder aller 188 Byte angehalten werden können, damit der RS-Encoder seine sechs Paritätsbytes einschieben kann. Die Spezifikation jedes Einzelmoduls ist im Git-Repository zu finden.

Berechnung Bitrate des MPEG2-TS

- gegeben: Symbolrate 4,5 MSym/s
- QPSK, also 2 Bit pro Symbol
  - aber: aus Faltungskodierer kommen 2 Bit pro Datenbit
  - d.h.: 2×4,5Mbit/s Datenstrom am Ausgang
- Durch Puncturing: Weglassen von Datenbits, damit geringere Bitrate
  - z.B. 2/3
    - 3 Ausgangsbits pro 2 Datenbits
    - Redundanz bedeutet Faktor 2
    - also: pro Datenbit 0,75 Ausgangsbits
    - 4,5 Mbit/s / 0,75 = 6 MBit/s
- RS erzeugt aus 188 Byte immer 204 Byte
  - Geringere Nutzdatenrate, Faktor 188/204 = 0,921…
  - 6 Mbit/s * 0,921 = 5,529 Mbit/s

- Also Gesamtechnung: Sendebitrate / Bit pro Symbol / Puncturing factor * RS-Faktor

Bei Weglassen des Puncturing (Code Rate 1/2) ist die Sendesymbolrate (=Bitrate nach RS) ein ganzzahliger Teiler der Systemfrequenz. Die Bitrate des TS errechnet sich nur durch Multiplikation mit dem Reed-Solomon-Overhead-Faktor. Folgende Tabelle fasst erreichbare Datenraten bei 50MHz Systemtakt zusammen.

Clock-Divider	Brutto-Datenrate	Netto-(TS)-Datenrate
1	50 MSym/s	46,08 MBit/s
2	25 MSym/s	23,04 MBit/s
3	16,67 MSym/s	15,36 MBit/s
4	12,5 MSym/s	11,51 MBit/s
5	10 MSym/s	9,22 MBit/s
6	8,33 MSym/s	7,68 MBit/s
7	7,14 MSym/s	6,58 MBit/s
8	6,25 MSym/s	6,76 MBit/s

Netzwerkprotokoll

Eine aufwendige Stack-Implementierung soll vermieden werden - diese Aufgabe könnte später mal ein IP-Stack übernehmen. Im Moment ist es wichtig, eine stabile, Datenflusskontrollierte Verbindung in den FPGA aufzubauen. Es wird daher eine einfache Zwei-Wege-Kommunikation definiert.

Allgemeiner Paketaufbau:

Ethernet-Header	IP-Header	UDP-Header	UDP-Daten	Ethernet-Footer

Folgende Anforderungen werden spezifiziert:

- Im Ethernet-Header sollen beliebige Quellen- und Ziel-MAC-Adresse stehen dürfen
- Im IP-Header sollen beliebige Ziel- und Quell-Adressen stehen dürfen - kein Feld wird überprüft
- Im UDP-Header wird der Ziel-Port überprüft. Die Länge kann optional überprüft werden
- Das Längen-Feld aller Protokollebenen wird ignoriert
- Zum Senden werden feste (vorher definierte) Pakete mit vorher berechneten Checksummen genutzt.

Pakete PC -> FPGA

- UDP-Pakete, Ziel-Port 40000
- Payload: 7*188 Byte Daten (7 MPEG-Frames)
    - optional: variable Länge, kann vom FPGA überprüft werden

Pakete FPGA -> PC

- UDP-Pakete, Ziel-Port 40001
- Payload: 1 Byte
    - 0x00: FIFO hat unteren Schwellwert erreicht - Datenrate erhöhen
    - 0x01: FIFO hat oberen Schwellwert erreicht - Datenrate erniedrigen

- 0x02: (optional) FIFO ist leer, einmalig
- 0x03: (optional) FIFO ist voll, einmalig

Pakete werden nicht wiederholt, bevor die entsprechende Bedingung nicht verlassen wurde (FIFO wieder auf normalen Füllzustand zurückgekehrt).