Übung zur Vorlesung Architekturen und Entwurf von Rechnersystemen

Prof. Dr-Ing. A. Koch
Jaco Hofmann, MSc.

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 16/17
Übungsblatt 4

Ab dieser Übung wird ein kleines System on Chip (SoC) entwickelt. Dabei lernen Sie Techniken wie Direct-Memory-Access (DMA) und Bussysteme in der Praxis kennen. Als Einstieg in diesen Themenkomplex wird in dieser Übung ein Stream-basierter Bildfilter entwickelt.

Aufgabe 4.1 Bildfilter mit ClientServer Interface

Aus den Vorlesungen kennen Sie bereits generische Interfaces wie das `GetPut` Interface. Damit die Bildfilter, die in den kommenden Übungen entwickelt werden, modular und einheitlich aufgebaut sind, benutzen wir ein solches generisches Interface.

```
1  typedef Bit#(8) Color;
2  typedef Bit#(8) GrayScale;
3
4  typedef struct {
5          Color r;
6          Color g;
7          Color b;
8  } RGB deriving(Bits, Eq);
9   module mkGray(Server#(RGB, GrayScale));
10  ...
11  endmodule
```

Die Details zum `Server` Interface finden Sie im BSV-Reference-Guide oder in BSV-by-example.

Implementieren Sie ein Modul, das RGB-Pixel eines Bildes über das `Server` Interface erhält und verwandeln Sie diesen Pixel in Grauwerte. Verwenden Sie dazu die Methode, die unter "Luma coding in video systems" auf https://en.wikipedia.org/wiki/Grayscale beschrieben wird:

$$Y = 0.299R + 0.587G + 0.114B$$

Die Größe von Floating-Point Einheiten in Hardware macht diese uninteressant in vielen Anwendungen. Anstatt von Floating-Point wird ein Q8.8 Fixed-Point Format für die Berechnung verwendet. Information über "Q"-Floating-Point finden Sie auf https://en.wikipedia.org/wiki/Q_(number_format). Gibt es in den Bluespec Libraries Unterstützung für arithmetische Operation auf diesem Format?

Aufgabe 4.2 Testen des Moduls

Testen Sie das von Ihnen entwickelte Modul mithilfe einer klassischen Testbench und zusätzlich mit BlueCheck.