# Übung zur Vorlesung Einführung in Computer Microsystems

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

TECHNISCHE UNIVERSITÄT DARMSTADT

**Sommersemester 15**
Übungsblatt 5

Diese Übung behandelt die Implementierung eines „Streaming Image Processor“ zur Reduzierung der Auflösung eines Eingabebildes. Ein Eingabebild der 2160p Auflösung 3840 × 2160 würde ein Ausgabebild der Auflösung 1080p 1920 × 1080 produzieren (Halbierung der Auflösung Horizontal und Vertikal).

Das Interface des Streaming Prozessors erwartet auf der einen Seite Pixeldaten in einem geeigneten Format wie 8 bit pro Farbe RGB. Die Ausgabeseite gibt Pixeldaten in eben jenem Format aus. Des weiteren soll die Breite einer Zeile des Ursprungsbildes über eine Methode auf beliebige gerade Auflösungen bis 2160p veränderbar sein.

## Aufgabe 5.1 Datentypen

Definieren Sie mit Hilfe von `typedef` einen geeigneten Datentyp für die Pixeldaten des Prozessors. Benutzen Sie entweder RGB(A) oder YCbCr Formate.

## Aufgabe 5.2 Interface

Implementieren Sie ein Interface für den oben beschriebenen Prozessor. Nutzen Sie wenn möglich standardisierte Interfaces (z.B. `ClientServer`).

## Aufgabe 5.3 Zwischenspeicherung der Eingabedaten

Überlegen Sie sich eine Möglichkeit die Eingabepixel bis zur Bearbeitung zu speichern. Versuchen Sie zu jeder Zeit nicht mehr als zwei Zeilen des Eingabebildes zu Speichern. Benutzen Sie zur Zwischenspeicherung das Package `RegFile`. Die Zeilenbreite soll bis 2160p einstellbar sein.

## Aufgabe 5.4 Ausgabebild

Berechnen Sie das Ausgabebild aus dem Eingabebild indem Sie den Durchschnitt der Farbwerte der entsprechenden Pixel des Eingabebildes berechnen. Diese sind die Pixel die am nächsten zum neuen Pixelwert gelegen sind. Für den Pixel $P_O(0,0)$ des Ausgabebildes benötigt man die Pixel $P_I(0,0), P_I(0,1), P_I(1,0), P_I(1,1)$ des Eingabebildes

$$P_O(0,0) = \frac{P_I(0,0) + P_I(0,1) + P_I(1,0) + P_I(1,1)}{4} \tag{1}$$

Geben Sie dieses Ausgabebild über das von Ihnen erstellte Interface aus.

## Aufgabe 5.5 Testbench

Testen Sie Ihr Modul mit Hilfe von Bluespec und einfachen Testdaten.

## Aufgabe 5.6 (BONUS: Leicht)

Mit Hilfe des BRAM Moduls können Sie echte Bilder in Ihrer Testbench verwenden.

```
1 BRAM_Configure cfg = defaultValue ;
2 //declare variable cfg
3 cfg.memorySize = 1024*32 ; //new value for memorySize
4 cfg.loadFormat = tagged Hex "ram.txt"; //value for loadFormat
5 BRAM2Port#(UInt#(15), Bit#(16)) bram <- mkBRAM2Server (cfg) ;
6 //instantiate 32K x 16 bits BRAM module
```

Die Eingabedaten werden entsprechend als Hexadezimalwerte in der Datei „ram.txt“ erwartet. Schaffen Sie eine Möglichkeit (z.B. Python Script) ein Bild in das erwartete Format umzuwandeln.

Die Ausgabe kann zum Beispiel über die `$display` Funktion erfolgen.

### Aufgabe 5.7 (BONUS: Mittel) C Code Testbench

Sie können C Funktionen in Ihre Testbench einbinden. Dieser Vorgang ist im Bluespec Reference Guide ab Kapitel 16 (ab Seite 146) beschrieben. Lesen Sie Bilder direkt über eine C Funktion ein und geben Sie das verkleinerte Bild über eine Grafikschnittstelle wie SDL aus.

### Aufgabe 5.8 (BONUS: Schwer) Bicubic Resampling

Ein besserer Algorithmus zur Auflösungsreduktion ist das Bicubic Resampling. Tauschen Sie das oben verwendete Verfahren durch Bicubic Resampling aus wie es zum Beispiel auf `https://clouard.users.greyc.fr/Pantheon/experiments/rescaling/index-en.html#bicubic` oder `https://code.google.com/a/eclipselabs.org/p/bicubic-interpolation-image-processing/source/browse/trunk/libimage.c` beschrieben ist. Zur Implementierung können Sie die Floating-Point Implementation aus `$BLUESPECDIR/BSVSource/Math/FloatingPoint.bsv` (Achtung! Keine Dokumentation.) verwenden.